FIGURE 1 — READ DATA FROM MEMORY OR PERIPHERALS

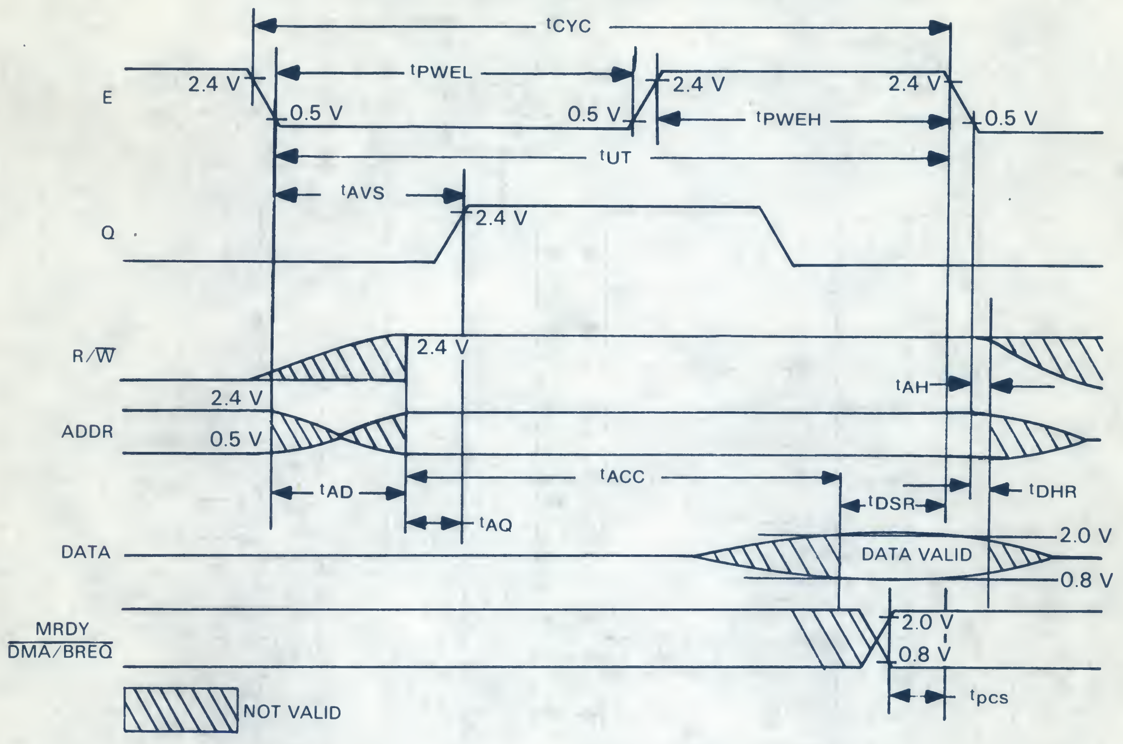


FIGURE 2 — WRITE DATA TO MEMORY OR PERIPHERALS

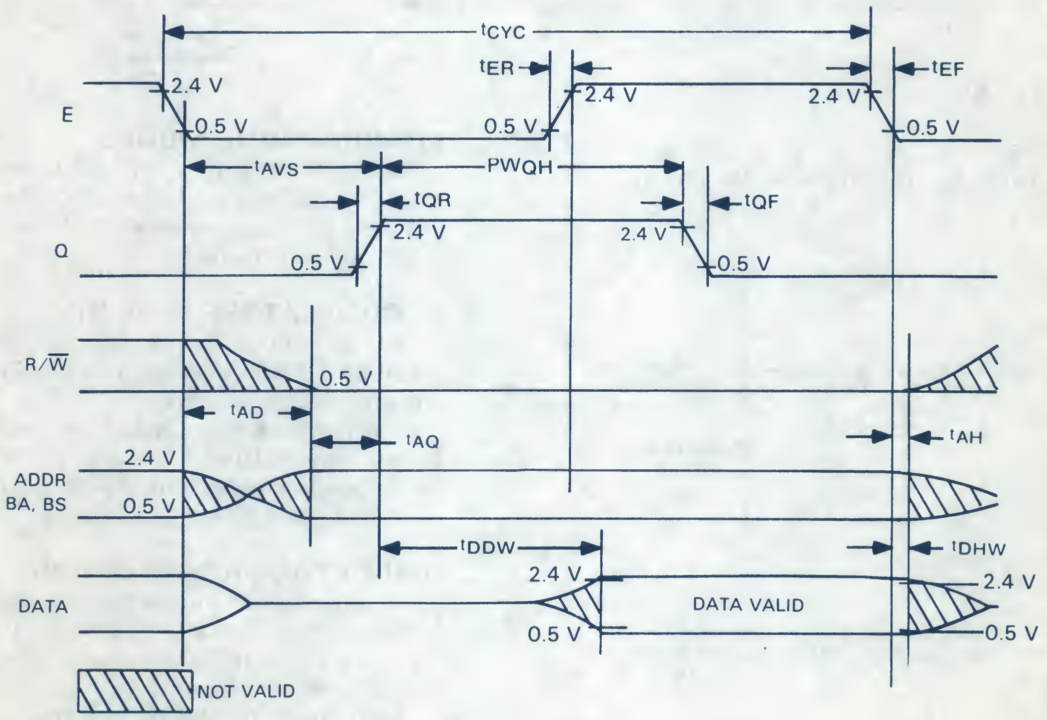


FIGURE 3 — MC6809 EXPANDED BLOCK DIAGRAM

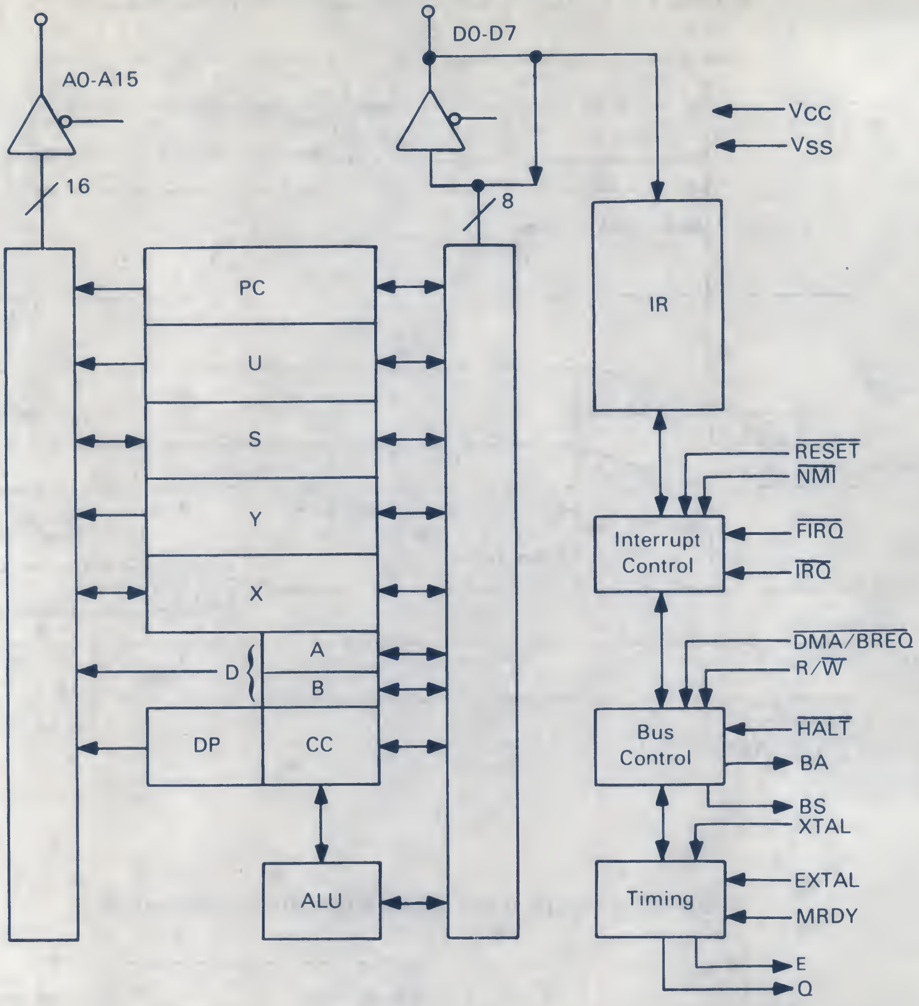
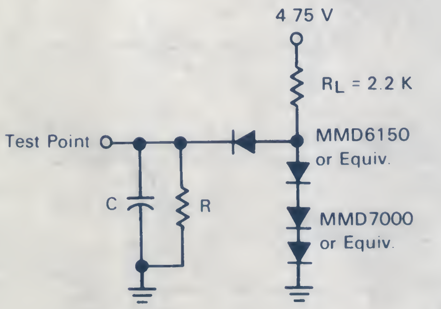


FIGURE 4 — BUS TIMING TEST LOAD



C = 30 pF for BA, BS	R = 11.7 kΩ for D0-D7
130 pF for D0-D7, E, Q	16.5 kΩ for A0-A15, E, Q
90 pF for A0-A15, R/W	24 kΩ for BA, BS

PROGRAMMING MODEL

As shown in Figure 5, the MC6809 adds three registers to the set available in the MC6800. The added registers include a direct page register, the User Stack pointer and a second Index Register.

ACCUMULATORS (A, B, D)

The A and B registers are general purpose accumulators which are used for arithmetic calculations and manipulation of data.

Certain instructions concatenate the A and B registers to form a single 16-bit accumulator. This is referred to as the D register, and is formed with the A register as the most significant byte.

DIRECT PAGE REGISTER (DP)

The Direct Page Register of the MC6809 serves to enhance the Direct Addressing Mode. The content of this register appears at the higher address outputs (A8-A15) during direct Addressing Instruction execution. This allows the direct mode to be used at any place in memory, under program control. To allow 6800 compatibility, all bits of this register are cleared during Processor Reset.

MOTOROLA Semiconductors

MC6809 (1.0 MHz)
MC68A09 (1.5 MHz)
MC68B09 (2.0 MHz)

Advance Information

8-BIT MICROPROCESSING UNIT

The MC6809 is a revolutionary high performance 8-bit microprocessor which supports modern programming techniques such as position independence, reentrancy, and modular programming.

This third-generation addition to the MC6800 family has major architectural improvements which include additional registers, instructions and addressing modes.

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The MC6809 has the most complete set of addressing modes available on any microprocessor today.

The MC6809 has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications.

MC6800 COMPATIBLE

- Hardware - Interfaces with All M6800 Peripherals
- Software - Upward Source Code Compatible Instruction Set and Addressing Modes

ARCHITECTURAL FEATURES

- Two 16-bit Index Registers
- Two 16-bit Indexable Stack Pointers
- Two 8-bit Accumulators can be concatenated to form one 16-bit Accumulator
- Direct Page Register allows Direct Addressing throughout memory map

HARDWARE FEATURES

- On chip oscillator (4 X fo XTAL)
- $\overline{\text{DMA/BREQ}}$ allows DMA operation or memory refresh
- Fast Interrupt Request Input stacks only Condition Code Register and Program Counter
- MRDY Input extends data access times for use with slow memory
- Interrupt Acknowledge output allows vectoring by devices
- SYNC Acknowledge output allows for synchronization to external event.
  - Single Bus-cycle RESET
  - Single 5-volt operation
  - NMI blocked after RESET until after first load of Stack Pointer
  - Early address valid allows use with slower memories

SOFTWARE FEATURES

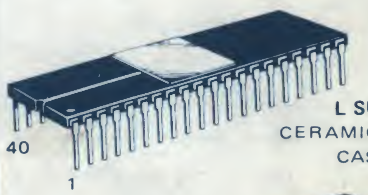
- 10 Addressing Modes
  - 6800 Upward compatible Addressing Modes
  - Direct Addressing anywhere in memory map
  - Long Relative Branches
  - Program Counter Relative
  - Indirection
  - Expanded Index Addressing
    - 0,5,8,16-bit constant offsets
    - 8, 16-bit accumulator offsets
    - Auto-increment/decrement
- Improved Stack Manipulation
- 1464 Instructions with unique addressing modes
- 8 x 8 unsigned multiply
- 16-bit arithmetic
- Transfer/Exchange all registers
- Push/Pull **any** or **all** registers
- Load Effective Address

This is advance information and specifications are subject to change without notice.

MOS

(N-CHANNEL, SILICON-GATE)

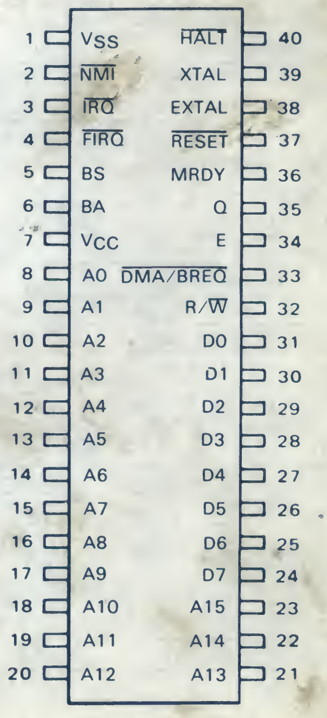
8-BIT MICROPROCESSING UNIT



L SUFFIX
CERAMIC PACKAGE
CASE 715

P SUFFIX
PLASTIC PACKAGE
CASE 711

PIN ASSIGNMENT



1 VSS — $\overline{\text{HALT}}$ 40
2 $\overline{\text{NMI}}$ — XTAL 39
3 $\overline{\text{IRQ}}$ — EXTAL 38
4 $\overline{\text{FIRQ}}$ — $\overline{\text{RESET}}$ 37
5 BS — MRDY 36
6 BA — Q 35
7 VCC — E 34
8 A0 — $\overline{\text{DMA/BREQ}}$ 33
9 A1 — R/$\overline{\text{W}}$ 32
10 A2 — D0 31
11 A3 — D1 30
12 A4 — D2 29
13 A5 — D3 28
14 A6 — D4 27
15 A7 — D5 26
16 A8 — D6 25
17 A9 — D7 24
18 A10 — A15 23
19 A11 — A14 22
20 A12 — A13 21

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 to +7.0	Vdc
Input Voltage	V_{in}	-0.3 to +7.0	Vdc
Operating Temperature Range	T_A	0 to +70	^{0}C
Storage Temperature Range	T_{stg}	-55 to +150	^{0}C
Thermal Resistance	θ_{JA}	70	^{0}C/W

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance circuit.

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 V ±5%, V_{SS} = 0, T_A = 0 to 70^{0}C unless otherwise noted.)

Characteristic		Symbol	Min	Typ	Max	Unit
Input High Voltage	Logic, EXtal $\overline{\text{RESET}}$	V_{IH}	V_{SS} + 2.0 V_{SS} + 4.0	— —	V_{DD} V_{DD}	Vdc
Input Low Voltage	Logic, EXtal, $\overline{\text{RESET}}$	V_{IL}	V_{SS} - 0.3	—	V_{SS} + 0.8	Vdc
Input Leakage Current (V_{in} = 0 to 5.25 V, V_{CC} = max)	Logic	I_{in}	—	1.0	2.5	μAdc
Output High Voltage (I_{Load} = -205 μAdc, V_{CC} = min) (I_{Load} = -145 μAdc, V_{CC} = min) (I_{Load} = -100 μAdc, V_{CC} = min)	D0-D7 A0-A15, R/$\overline{\text{W}}$, Q, E BA, BS	V_{OH}	V_{SS} + 2.4 V_{SS} + 2.4 V_{SS} + 2.4	— — —	— — —	Vdc
Output Low Voltage (I_{Load} = 2.0 mAdc, V_{CC} = min)		V_{OL}	—	—	V_{SS} +0.5	Vdc
Power Dissipation		P_D	—	—	1.0	W
Capacitance # (V_{in} = 0, T_A = 25^{0}C, f = 1.0 MHz)	D0-D7 Logic Inputs, EXtal A0-A15, R/W	C_{in} C_{out}	— — —	10 7 —	15 10 12	pF
Frequency of Operation (Crystal or External Input)	MC6809 MC68A09 MC68B09	f f_{XTAL} f_{XTAL}	— — —	— — —	4 6 8	MHz
Three-State (Off State) Input Current (V_{in} = 0.4 to 2.4 V, V_{CC} = max)	D0-D7 A0-A15, R/W	I_{TSI}	— —	2.0 —	10 100	μAdc

READ/WRITE TIMING (Reference Figures 1 and 2)

Characteristic	Symbol	MC6809			MC68A09			MC68B09			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Cycle Time	t_{CYC}	1000	—	—	667	—	—	500	—	—	ns
Total Up Time	t_{UT}	975	—	—	640	—	—	480	—	—	ns
Peripheral Read Access Time t_{ac} = (t_{AD} = t_{DSR})	t_{ACC}	695	—	—	440	—	—	320	—	—	ns
Data Setup Time (Read)	t_{DSR}	80	—	—	60	—	—	40	—	—	ns
Input Data Hold Time	t_{DHR}	10	—	—	10	—	—	10	—	—	ns
Output Data Hold Time	t_{DHW}	30	—	—	30	—	—	30	—	—	ns
Address Hold Time (Address, R/$\overline{\text{W}}$)	t_{AH}	30	—	—	30	—	—	30	—	—	ns
Address Delay	t_{AD}	—	—	200	—	—	140	—	—	110	ns
Data Delay Time (Write)	t_{DDW}	—	—	225	—	—	180	—	—	145	ns
E_{low} to Q_{high} Time	t_{AVS}	—	—	250	—	—	165	—	—	125	ns
Address Valid to Q_{high}	t_{AQ}	25	—	—	25	—	—	15	—	—	ns
Processor Clock Low	t_{PWEL}	450	—	—	295	—	—	210	—	—	ns
Processor Clock High	t_{PWEH}	450	—	—	280	—	—	220	—	—	ns
MRDY Set Up Time	t_{PCSR}	60	—	—	60	—	—	60	—	—	ns
Interrupts Set Up Time	t_{PCS}	200	—	—	140	—	—	110	—	—	ns
$\overline{\text{HALT}}$ Set Up Time	t_{PCSH}	200	—	—	140	—	—	110	—	—	ns
$\overline{\text{RESET}}$ Set Up Time	t_{PCSR}	200	—	—	140	—	—	110	—	—	ns
$\overline{\text{DMA/BREQ}}$ Set Up Time	t_{PCSD}	125	—	—	125	—	—	125	—	—	ns
Crystal Osc Start Time	t_{rc}	100	—	—	100	—	—	100	—	—	ms
E Rise and Fall Time	t_{ER}, t_{EF}	5	—	25	5	—	25	5	—	20	ns
Processor Control Rise/Fall	t_{PCR}, t_{PLF}	—	—	100	—	—	100	—	—	100	ns
Q Rise and Fall Time	t_{QR}, t_{QF}	5	—	25	5	—	25	5	—	20	ns
Q Clock High	t_{PWQH}	450	—	—	280	—	—	220	—	—	ns

FIGURE 5 — PROGRAMMING MODEL OF THE MICROPROCESSING UNIT

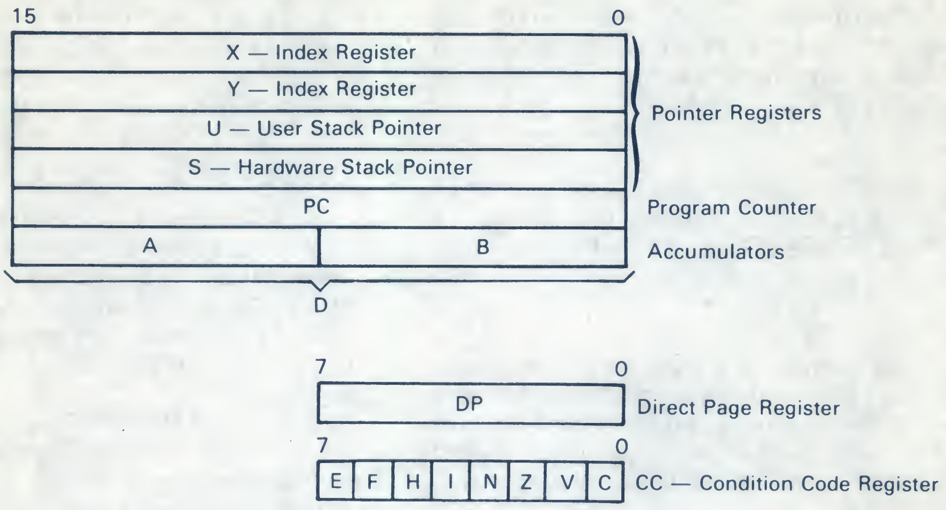
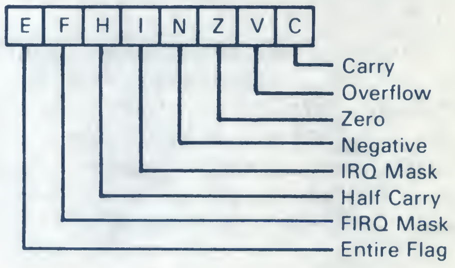


FIGURE 6 — CONDITION CODE REGISTER FORMAT



INDEX REGISTERS (X,Y)

The Index Registers are used in indexed mode of addressing. The 16-bit address in this register takes part in the calculation of effective addresses. This address may be used to point to data directly or may be modified by an optional constant or register offset. During some indexed modes, the contents of the index register are incremented and decremented to point to the next item of tabular type data. All four pointer registers (X,Y,U,S) may be used as index registers.

STACK POINTERS (U,S)

The Hardware Stack Pointer (S) is used automatically by the processor during subroutine calls and interrupts. The stack pointers of the MC6809 point to the top of the stack, in contrast to the MC6800 stack pointer, which pointed to the next free location on the stack. The User Stack Pointer (U) is controlled exclusively by the programmer thus allowing arguments to be passed to and from subroutines with ease. Both Stack Pointers have the same indexed mode addressing capabilities as the X and Y registers, but also support **Push** and **Pull** instructions. This allows the MC6809 to be used efficiently as a stack processor, greatly enhancing its ability to support higher level languages and modular programming.

PROGRAM COUNTER

The Program Counter is used by the processor to point to the address of the next instruction to be executed by the processor. Relative Addressing is provided allowing the Program Counter to be used like an index register in some situations.

CONDITION CODE REGISTER

The condition code register defines the State of the Processor at any given time, see Figure 6.

CONDITION CODE REGISTER DESCRIPTION

BIT 0 (C)

Bit 0 is the Carry Flag, and is usually the carry from the binary ALU. C is also used to represent a 'borrow' from subtract like instructions (CMP, NEG, SUB, SBC). Here the carry flag is the complement of the carry from the binary ALU.

BIT 1 (V)

Bit 1 is the overflow flag, and is set to a one by an operation which causes a signed two's complement arithmetic overflow. This overflow is detected in an operation in which the carry from the MSB in the ALU does not match the carry from the MSB-1.

BIT 2 (Z)

Bit 2 is the zero flag, and is set to a one if the result of the previous operation was identically zero.

BIT 3 (N)

Bit 3 is the negative flag, which contains exactly the value of the MSB of the result of the preceeding operation. Thus, a negative two's-complement result will leave N set to a one.

BIT 4 (I)

Bit 4 is the $\overline{\text{IRQ}}$ mask bit. The processor will not recognize interrupts from the IRQ line if this bit is set to a one. $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$, $\overline{\text{RESET}}$, and SWI all set I to a one; SWI2 AND SWI3 do not affect I.

BIT 5 (H)

Bit 5 is the half-carry bit, and is used to indicate a carry from bit 3 in the ALU as a result of an 8-bit addition only (ADC or ADD). This bit is used by the DAA instruction to perform a BCD decimal add adjust operation. The state of this flag is undefined in all subtract-like instructions.

BIT 6 (F)

Bit 6 is the $\overline{\text{FIRQ}}$ mask bit. The processor will not recognize interrupts from the FIRQ line if this bit is a one. $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, SWI, and $\overline{\text{RESET}}$ all set F to a one. $\overline{\text{IRQ}}$, SWI2 and SWI3 do not affect F.

BIT 7 (E)

Bit 7 is the entire flag, and when set to a one indicates that the complete machine state (all the registers) was stacked, as opposed to the subset state (PC and CC). The E bit of the stacked CC is used on a return from interrupt (RTI) to determine the extent of the unstacking. Therefore, the current E left in the Condition Code Register represents past action.

MC6809 MPU SIGNAL DESCRIPTION

POWER (V_{SS}, V_{CC})

Two pins are used to supply power to the part: V_{SS} is ground or 0 volts, while V_{CC} is +5.0 V ±5%.

ADDRESS BUS (A0-A15)

Sixteen pins are used to output address information from the MPU onto the Address Bus. When the processor does not require the bus for a data transfer, it will output address FFFF16, R/$\overline{\text{W}}$ = 1, and BS = 0. Addresses are valid on the rising edge of Q (see Figure 1 and 2). All address bus drivers are made high-impedance when output Bus Available (BA) is high. Each pin will drive one Schottky TTL load and typically 90 pF.

DATA BUS (D0-D7)

These eight pins provide communication with the system bi-directional data bus. Each pin will drive one Schottky TTL load and typically 130 pF.

READ/WRITE (R/$\overline{\text{W}}$)

This signal indicates the direction of data transfer on the data bus. A low indicates that the MPU is writing data onto the data bus. R/$\overline{\text{W}}$ is made high impedance when BA is high. R/$\overline{\text{W}}$ is valid on the rising edge of Q, refer to Figure 1 and 2.

$\overline{\text{RESET}}$

A low level on this Schmitt-trigger input for greater than one bus cycle will reset the MPU as shown Figure 7. The Reset vectors are fetched from locations FFFE16 and FFFF16 (Table 1) when Interrupt Acknowledge is true, (BA ∧ BS = 1). During initial power-on, the Reset line should be held low until the clock oscillator is fully operational; see Figure 8.

Because the MC6809 Reset pin has a Schmitt-trigger input with a threshold voltage higher than that of standard peripherals, a simple R/C network may be used to reset the entire system. This higher threshold voltage insures that all peripherals are out of the reset state before the Processor.

$\overline{\text{HALT}}$

A low level on this input pin will cause the MPU to stop running at the end of the present instruction and remain halted indefinitely without loss of data. When Halted, the BA output is driven high indicating the buses are high-impedance. BS is also high which indicates the processor is in the Halt or Bus Grant state. While halted, the MPU will not respond to external real-time requests ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) although $\overline{\text{DMA/BREQ}}$ will always be accepted, and $\overline{\text{NMI}}$ or $\overline{\text{RESET}}$ will be latched for later response. During the Halt state Q and E continue to run normally. If the MPU is not running ($\overline{\text{RESET}}$, $\overline{\text{DMA/BREQ}}$), a halted state (BA and BS = 1) can be achieved by pulling $\overline{\text{HALT}}$ low while $\overline{\text{RESET}}$ is still low. If $\overline{\text{DMA/BREQ}}$ and $\overline{\text{HALT}}$ are both pulled low, the processor will reach the last cycle of the instruction (by reverse cycle stealing) where the machine will then become halted. See Figure 9.

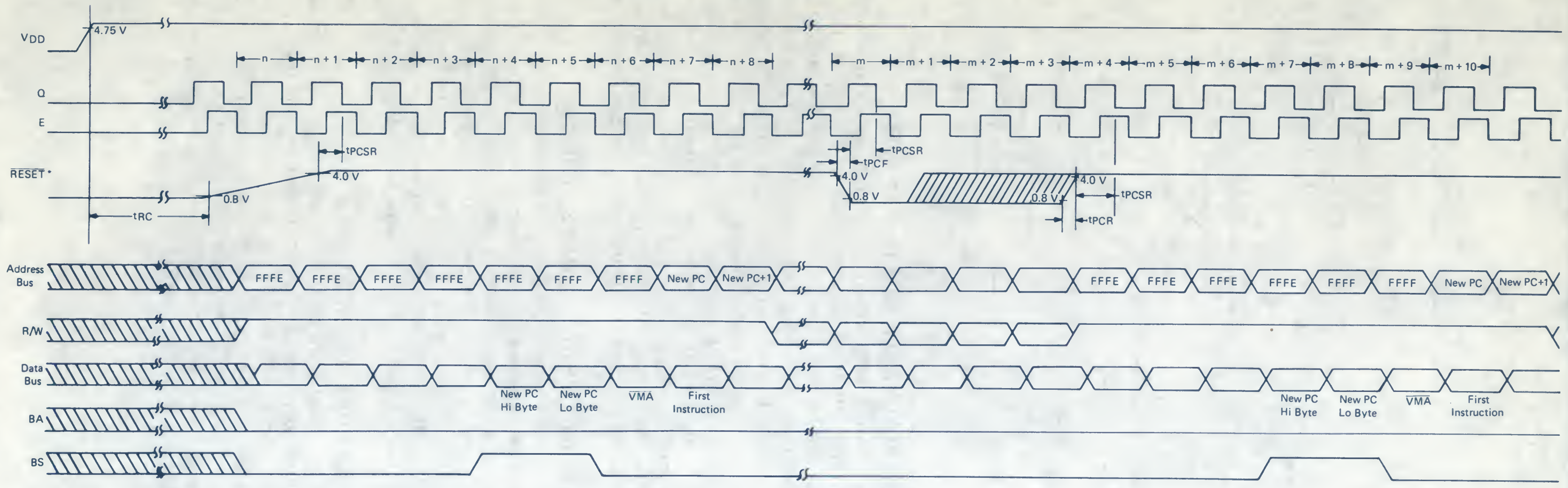
BUS AVAILABLE, BUS STATUS (BA, BS)

The Bus Available output is an indication of an internal control signal which makes the MOS buses of the MPU high-impedance. This signal does not imply that the bus will be available for more than one cycle. When BA goes low, an additional dead cycle will elapse before the MPU acquires the bus.

The Bus Status output signal, when decoded with BA, represents the MPU state (valid with leading edge of Q):

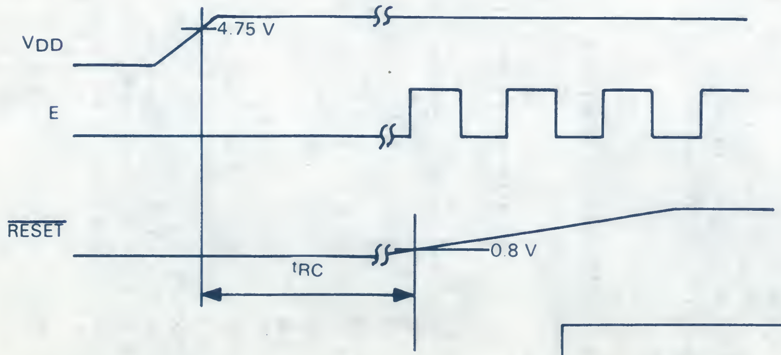
MPU State		
BA	BS	
0	0	Normal (Running)
0	1	Interrupt Acknowledge
1	0	SYNC Acknowledge
1	1	HALT or Bus Grant

FIGURE 7 — $\overline{\text{RESET}}$ TIMING

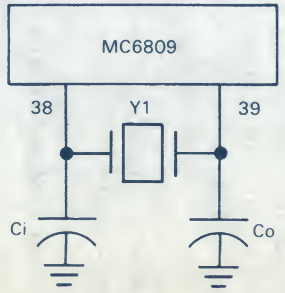


*Note: Parts with date codes prefixed by 7F will come out of Reset one cycle sooner than shown.

FIGURE 8 — CRYSTAL CONNECTIONS AND OSCILLATOR START UP



Y1	C_{in}	C_{out}
8 MHz	18 pF	18 pF
6 MHz	20 pF	20 pF
4 MHz	24 pF	24 pF



6809 Crystal Parameters*

	3.58 MHz	4.00 MHz	6.0 MHz	8.0 MHz
RS	60 Ω	50 Ω	30-50 Ω	20-40 Ω
C_0	3.5 pF	6.5 pF	4-6 pF	4-6 pF
C_1	.015 pF	.025 pF	.01-.02 pF	.01-.02 pF
C_{in}, C_{out}	25 pF	25 pF	25 pF	25 pF
Q	40 K	30 K	≈20 K	≈20 K

All Parameters Are ±10%

*Note: These are representative AT-cut crystal parameters only. Crystals of other types of cut that work may also be used.

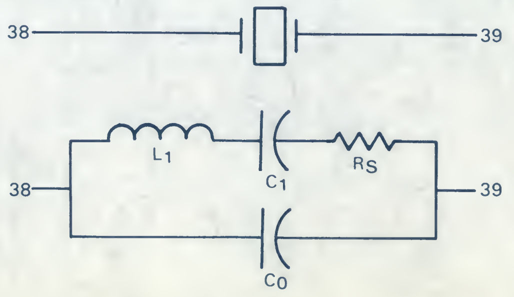
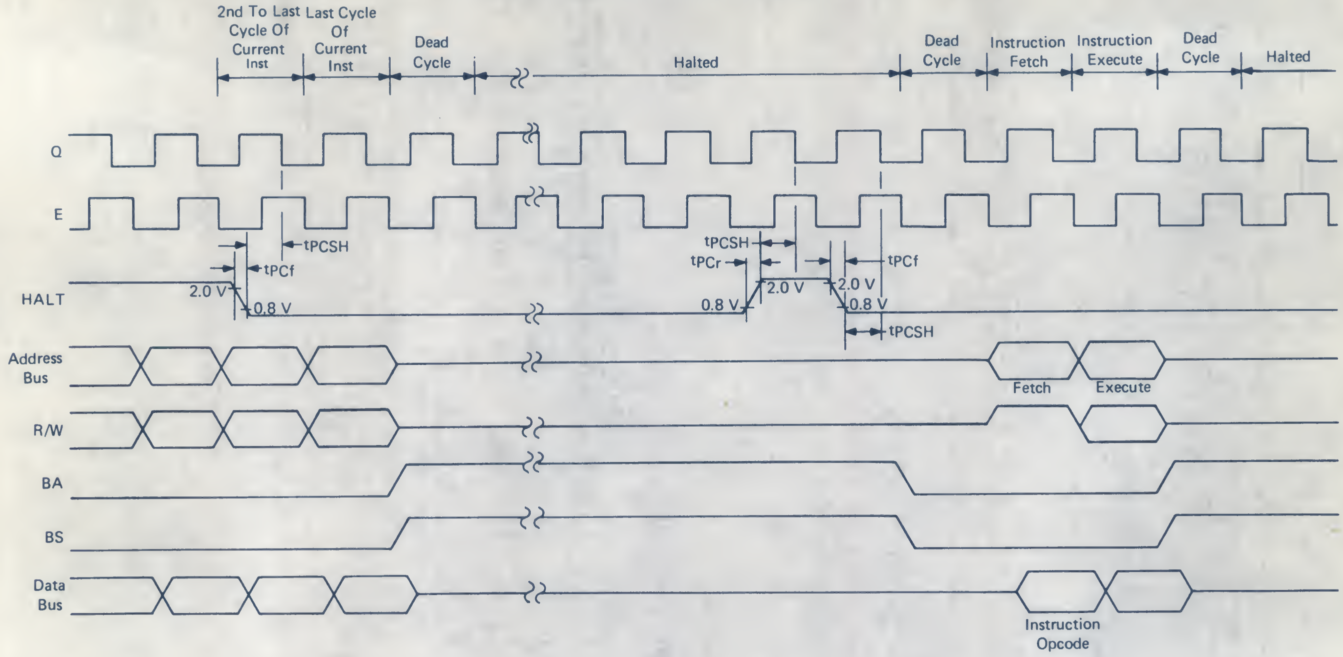


FIGURE 9 — $\overline{\text{HALT}}$ AND SINGLE INSTRUCTION EXECUTION FOR SYSTEM DEBUG



Interrupt Acknowledge is indicated during both cycles of a hardware-vector-fetch ($\overline{\text{RESET}}$, $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$, SWI, SWI2, SWI3). This signal, plus decoding of the lower 4 address lines can provide the user with an indication of which interrupt level is being serviced and allow vectoring by device, (see Table 1).

Sync Acknowledge is indicated while the MPU is waiting for external synchronization on an interrupt line.

Halt/Bus Grant is true when the MC6809 is in a Halt or Bus Grant condition.

TABLE 1: MEMORY MAP FOR INTERRUPT VECTORS

Memory Map For Vector Location		Interrupt Vector Description
MS	LS	
FFFE	FFFF	$\overline{\text{RESET}}$
FFFC	FFFD	$\overline{\text{NMI}}$
FFFA	FFFB	SWI
FFF8	FFF9	$\overline{\text{IRQ}}$
FFF6	FFF7	$\overline{\text{FIRQ}}$
FFF4	FFF5	SWI2
FFF2	FFF3	SWI3
FFF0	FFF1	Reserved

***NOTE:** $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$ and $\overline{\text{IRQ}}$ requests are latched by the falling edge of every Q except during cycle stealing operations (e.g., DMA) where only $\overline{\text{NMI}}$ is latched. From this point, a delay of at least one bus cycle will occur before the interrupt is serviced by the MPU.

NON MASKABLE INTERRUPT ($\overline{\text{NMI}}$)

A negative **edge** on this input requests that a non-maskable interrupt sequence be generated. A non-maskable interrupt cannot be inhibited by the program, and also has a higher priority than $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$ or software interrupts. During recognition of an $\overline{\text{NMI}}$, the entire machine state **is** saved on the hardware stack. After reset, an NMI will not be recognized until the first program load of the Hardware Stack Pointer (S). The pulse width of NMI low must be at least one E cycle. If the $\overline{\text{NMI}}$ input does not meet the minimum set up with respect to Q, the interrupt will not be recognized until the next cycle. See Figure 10.

FAST-INTERRUPT REQUEST ($\overline{\text{FIRQ}}$)

A low level on this input pin will initiate a fast interrupt sequence, provided its mask bit (F) in the CC is clear. This sequence has priority over the standard Interrupt Request ($\overline{\text{IRQ}}$), and is fast in the sense that it stacks only the contents of the condition code register and the program counter. The interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 11.

INTERRUPT REQUEST ($\overline{\text{IRQ}}$)

A low **level** input on this pin will initiate an Interrupt Request sequence provided the mask bit (I) in the CC is clear. Since $\overline{\text{IRQ}}$ stacks the entire machine state it provides a slower response to interrupts than $\overline{\text{FIRQ}}$. $\overline{\text{IRQ}}$ also has a lower priority than $\overline{\text{FIRQ}}$. Again, the interrupt service rouine should clear the source of the interrupt before doing an RTI. See Figure 10.

FIGURE 10 — $\overline{\text{IRQ}}$ AND $\overline{\text{NMI}}$ INTERRUPT TIMING

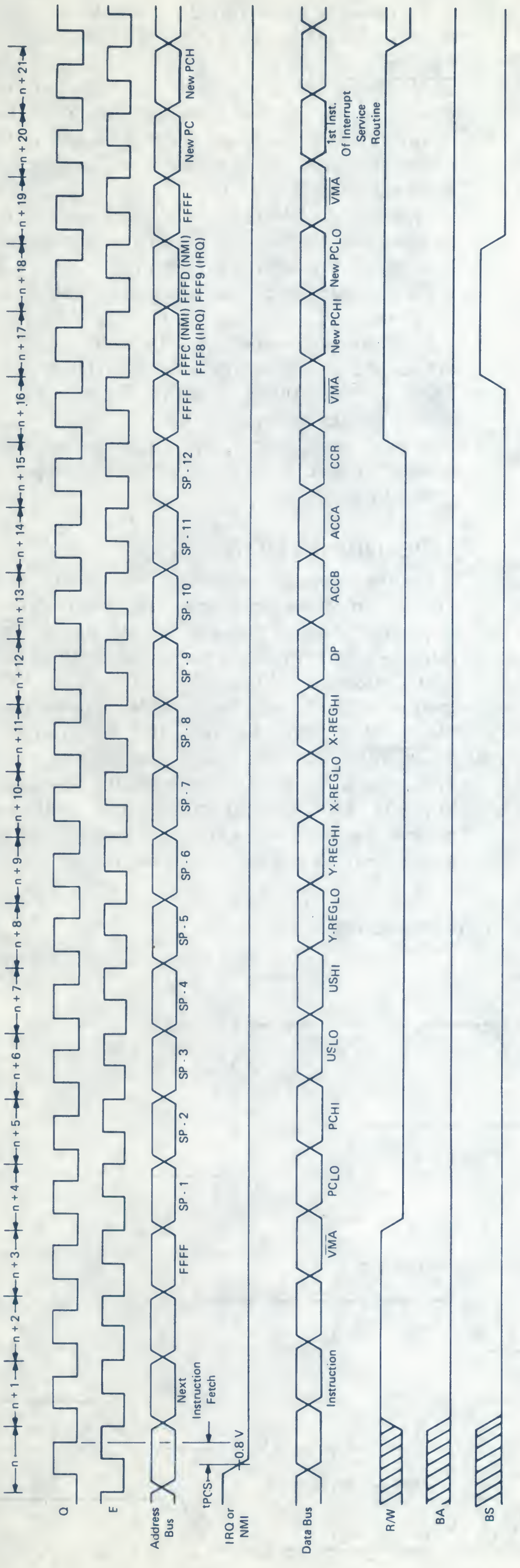
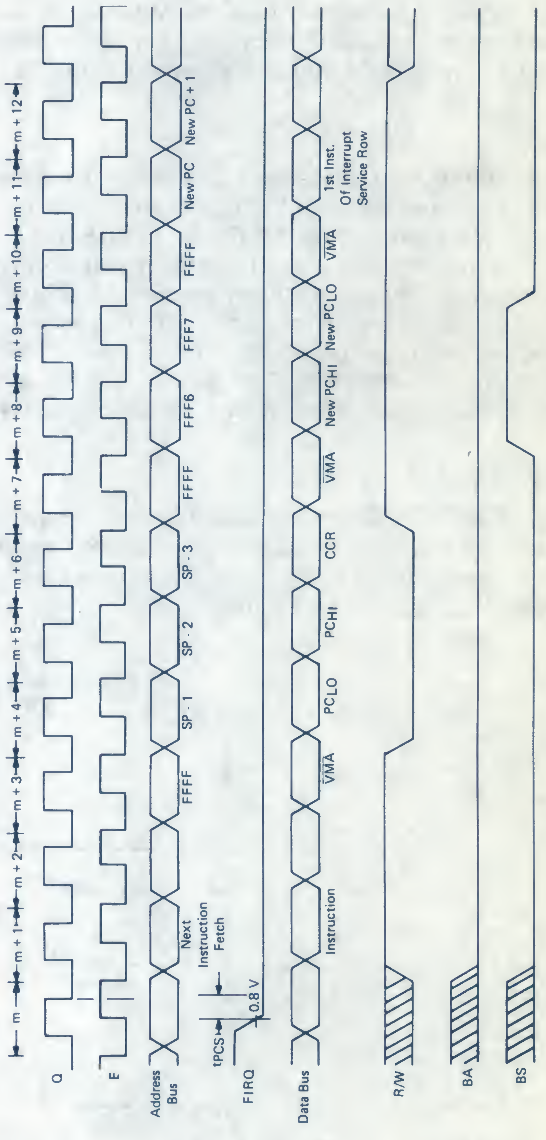


FIGURE 11 — $\overline{\text{FIRQ}}$ INTERRUPT TIMING



XTAL, EXTAL

These input pins are used to connect the on-chip oscillator to an external parallel-resonant crystal. Alternately, the pin EXTAL may be used as a TTL level input for external timing by grounding XTAL. The crystal or external frequency is 4 times the bus frequency, see Figure 8. Proper RF layout techniques should be observed in the layout of printed circuit boards.

E, Q

E is similar to the MC6800 bus timing signal $\phi 2$; Q is a quadrature clock signal which leads E. Q has no parallel on the MC6800. Addresses from the MPU will be valid with the leading edge of Q. Data is latched on the falling edge of E. Timing for E and Q is shown in Figure 12.

MRDY

This input control signal allows stretching of E to extend data-access time. When MRDY is high, E will be in normal operation. When MRDY is low, E may be stretched integral multiples of quarter (¼) bus cycles, thus allowing interface to slow memories as shown in Figure 13. A maximum stretch is 10 microseconds. During non-valid memory accesses ($\overline{\text{VMA}}$ cycles), MRDY has no effect on stretching E. This inhibits slowing the processor speed during "don't care" bus accesses.

$\overline{\text{DMA/BREQ}}$

The $\overline{\text{DMA/BREQ}}$ input provides a method of suspending execution and acquiring the MPU bus for another use as shown in Figure 14. Typical uses include DMA and dynamic memory refresh.

Transition of $\overline{\text{DMA/BREQ}}$ should occur during Q. A low level on this pin will stop instruction execution at the end of the current cycle. The MPU will acknowledge $\overline{\text{DMA/BREQ}}$ by setting BA and BS to a one. The requesting device will now have up to 15 bus cycles before the MPU retrieves the bus for self-refresh. Self-refresh requires one bus cycle with a leading and trailing dead cycle, see Figure 15.

Typically, the DMA controller will request to use the bus by asserting the $\overline{\text{DMA/BREQ}}$ pin low on the leading edge of E. When the MPU replies with BA = BS = 1, that cycle will be a dead cycle used to transfer control to the DMA controller.

False memory accesses should be prevented during any dead cycles. When BA is cleared (either as a result of $\overline{\text{DMA/BREQ}}$ = HIGH or MPU self-refresh), the DMA device should be taken off the bus.

Another dead cycle will elapse before the MPU is allowed a memory access to transfer control without contention.

MPU OPERATION

During normal operation, the MPU fetches an instruction from memory and then executes the requested function. This sequence begins at $\overline{\text{RESET}}$ and is repeated indefinitely unless altered by a special instruction or hardware occurrence. Software instructions that alter normal MPU operation are: SWI, SWI2, SWI3, CWAI, RTI and SYNC. An interrupt, $\overline{\text{HALT}}$ or $\overline{\text{DMA/BREQ}}$ can also alter the normal execution of instructions. Figure 16 illustrates the flow chart for the MC6809. The left-half of the flow chart represents normal operation; the right-half represents the flow when an interrupt or special instruction occurs.

FIGURE 12 — E/Q RELATIONSHIP

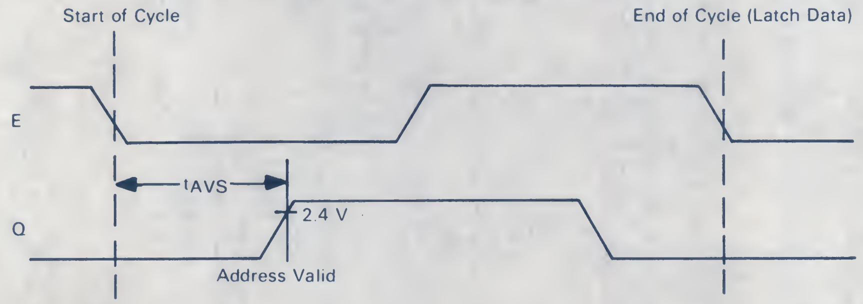


FIGURE 13 — MRDY TIMING

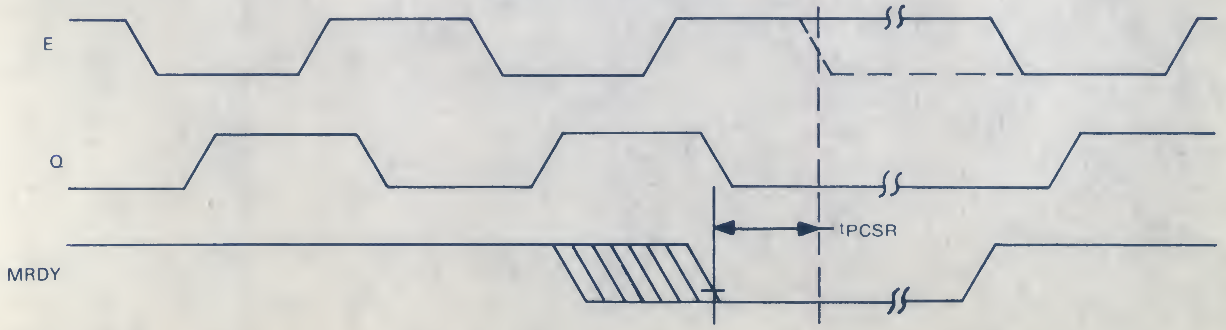
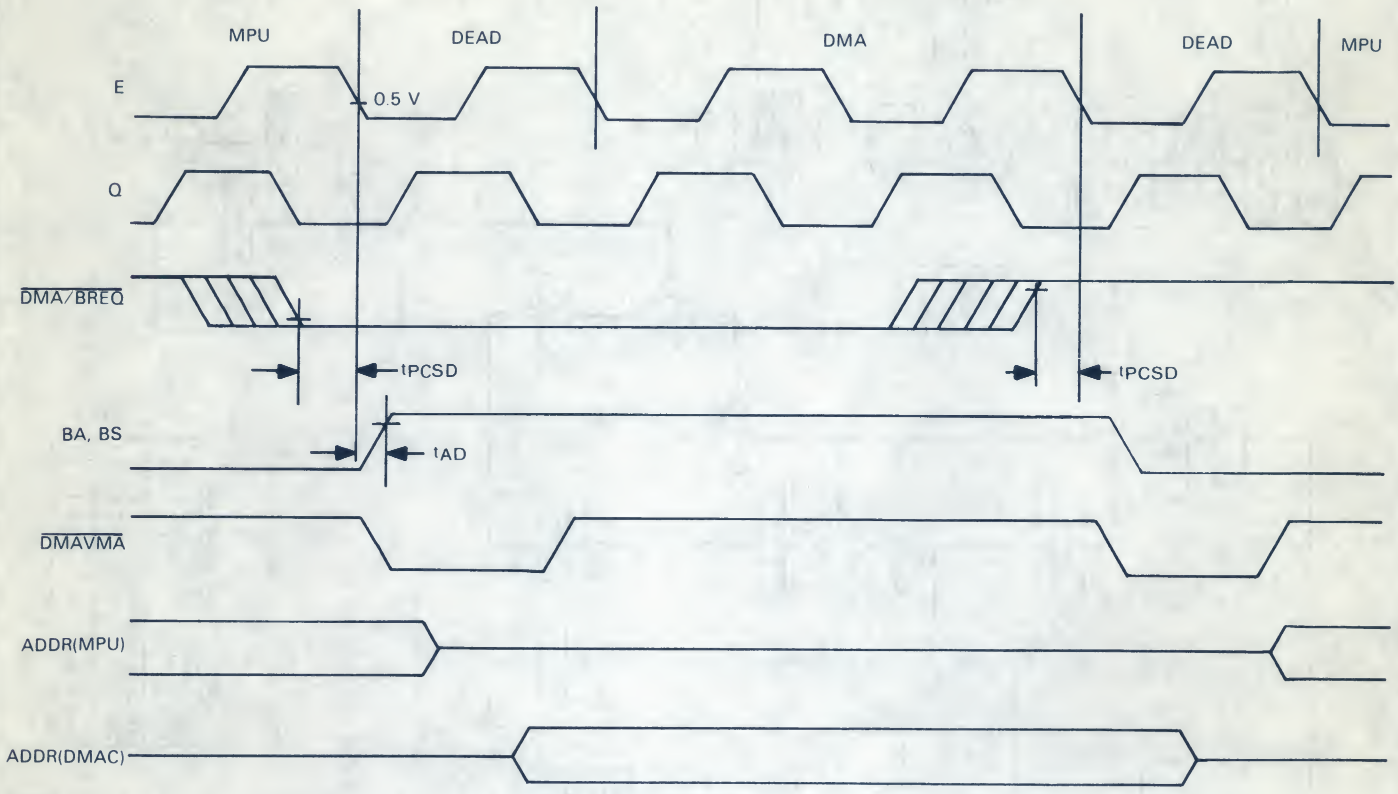


FIGURE 14 — TYPICAL DMA TIMING (<14 CYCLES)



NOTE:
DMAVMA is a signal which is developed externally, but is a system requirement for DMA.

FIGURE 15 — AUTO-REFRESH DMA TIMING (>14 CYCLES)

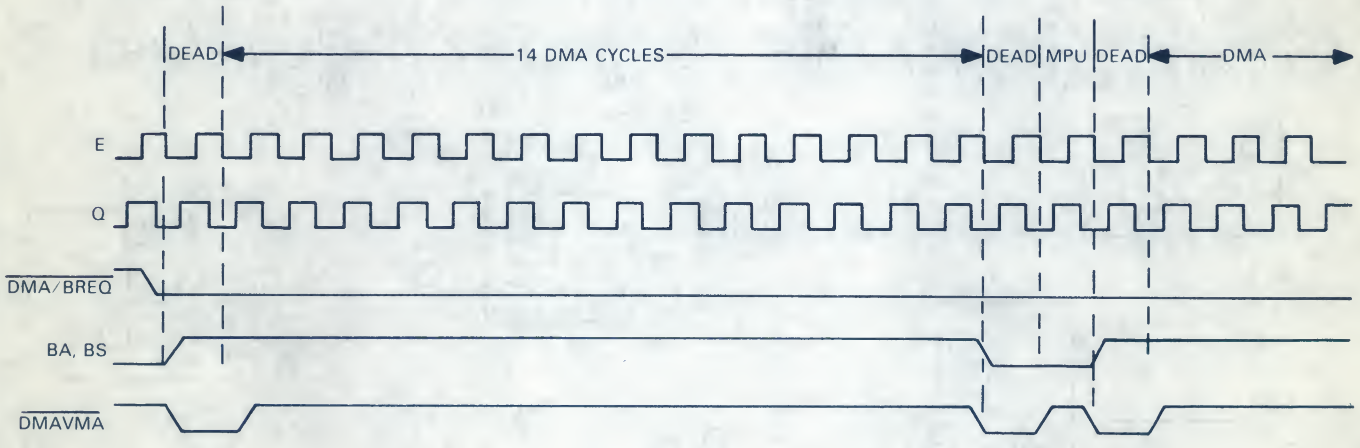
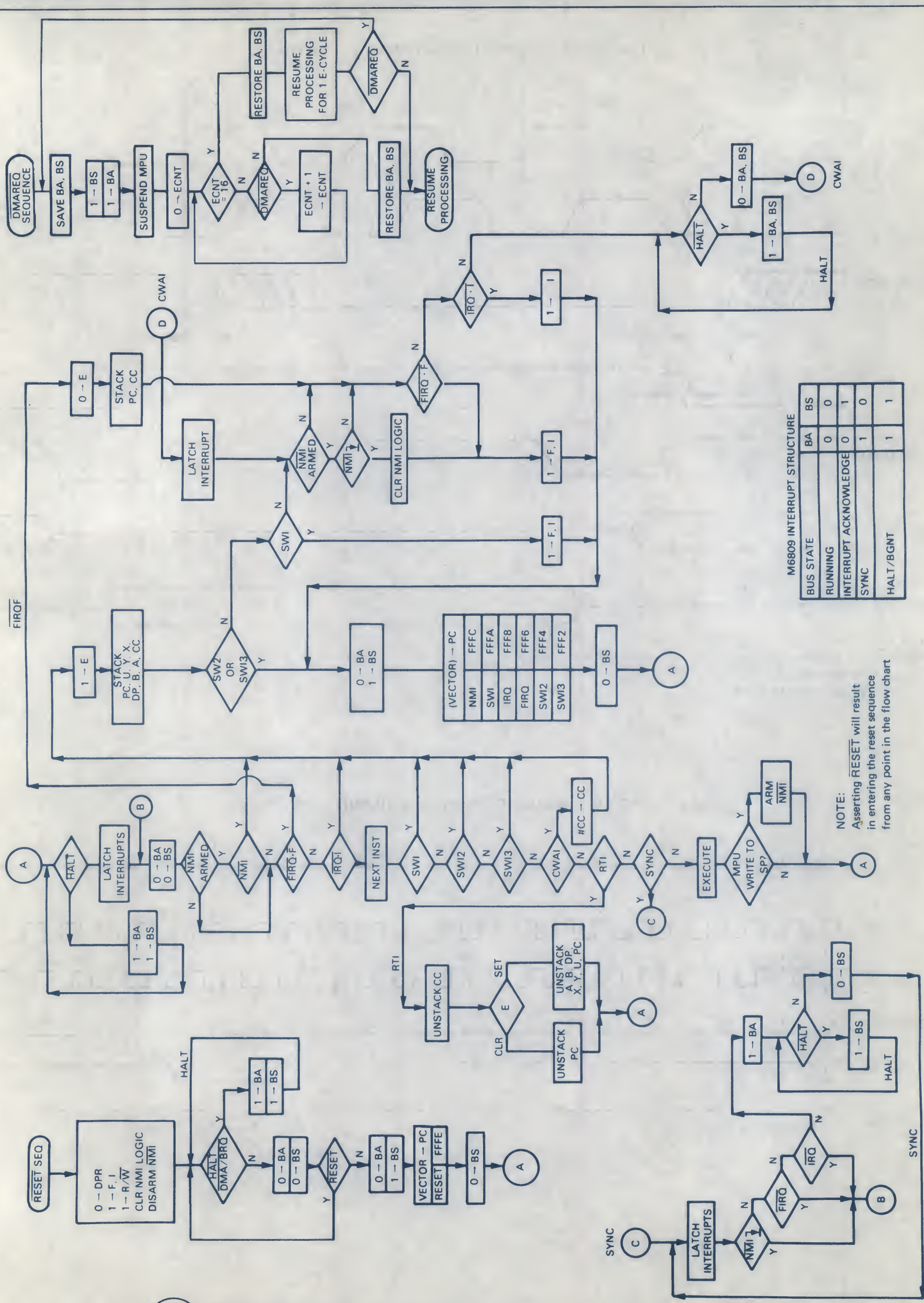


FIGURE 16 — MPU FLOWCHART



M6809 INTERRUPT STRUCTURE

BUS STATE	BA	BS
RUNNING	0	0
INTERRUPT ACKNOWLEDGE	0	1
SYNC	1	0
HALT/BGNT	1	1

NOTE:
Asserting RESET will result in entering the reset sequence from any point in the flow chart

ADDRESSING MODES

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The MC6809 has the most complete set of addressing modes available on any microcomputer today. For example, the MC6809 has 59 basic instructions, however it recognizes 1464 different variations of instructions and addressing modes. The new addressing modes support modern programming techniques. The following addressing modes are available on the MC6809:

- Inherent (Includes Accumulator)
- Immediate
- Extended
  - Extended Indirect
- Direct
- Register
- Indexed
  - Zero-Offset
  - Constant Offset
  - Accumulator Offset
  - Auto Increment/Decrement
  - Indexed Indirect
- Relative
  - Short/Long Relative Branching
  - Program Counter Relative Addressing

INHERENT (INCLUDES ACCUMULATOR)

In this addressing mode, the opcode of the instruction contains all the address information necessary. Examples of Inherent Addressing are: ABX, DAA, SWI, ASRA, and CLRB.

IMMEDIATE ADDRESSING

In Immediate Addressing, the effective address of the data is the location immediately following the opcode; the data to be used in the instruction immediately follows the opcode of the instruction. The MC6809 uses both 8 and 16-bit immediate values depending on the size of argument specified by the opcode. Examples of instructions with Immediate Addressing are:

```
LDA #$20
LDX #$F000
LDY #CAT
```

Note: # signifies Immediate addressing, $ signifies hexadecimal value

EXTENDED ADDRESSING

In Extended Addressing the contents of the two bytes immediately following the opcode fully specify the 16-bit effective address used by the instruction. Note that the address generated by an extended instruction defines an absolute address and is not position independent. Examples of Extended Addressing include:

```
LDA CAT
STX MOUSE
LDD $2000
```

EXTENDED INDIRECT

As a special case of indexed addressing (discussed below), one level of indirection may be added to Extended Addressing. In Extended Indirect, the two bytes following the postbyte of an Indexed instruction contains the address of the address of the data.

```
LDA [CAT]
LDX [$FFFE]
STU [DOG]
```

DIRECT ADDRESSING

Direct addressing is similar to extended addressing except that only one byte of address follows the opcode. This byte specifies the lower 8 bits of the address to be used. The upper 8 bits of the address are supplied by the direct page register. Since only one byte of address is required in direct addressing, this mode requires less memory and executes faster than extended addressing. Of course, only 256 locations (one page) can be accessed without redefining the contents of the DP register. Since the DP register is set to $00 on Reset, direct addressing on the MC6809 is compatible with direct addressing on the M6800. Indirection is not allowed in direct addressing. Some examples of direct addressing are:

```
LDA     $30
SETDP   $10 (Assembler directive)
LDB     $1030
LDD     < CAT
```

Note: < is an assembler directive which forces direct addressing.

REGISTER ADDRESSING

Some opcodes are followed by a byte that defines a register or set of registers to be used by the instruction, this is called a POSTBYTE. Some examples of register addressing are:

```
TFR     X,Y         Transfers X into Y
EXG     A,B         Exchanges A with B
PSHS    A,B,X,Y     Push onto S Y,X,B, then A
PULU    X,Y,D       Pull from U D,X, then Y
```

INDEXED ADDRESSING

In all indexed addressing one of the pointer registers (X, Y, U, S, and sometimes PC) is used in a calculation of the effective address of the operand to be used by the instruction. Five basic types of indexing are available and are discussed below. The postbyte of an indexed instruction specifies the basic type and variation of the addressing mode as well as the pointer register to be used. Figure 17 lists the legal formats for the postbyte. Table 2 gives the assembler form and the number of cycles and bytes added to the basic values for indexed addressing for each variation.

FIGURE 17 — INDEXED ADDRESSING POSTBYTE REGISTER BIT ASSIGNMENTS

Post-Byte Register Bit								Indexed Addressing Mode
7	6	5	4	3	2	1	0	
0	R	R	X	X	X	X	X	EA = ,R ± 4 Bit Offset
1	R	R	0	0	0	0	0	,R+
1	R	R	I	0	0	0	1	,R++
1	R	R	0	0	0	1	0	,-R
1	R	R	I	0	0	1	1	,--R
1	R	R	I	0	1	0	0	EA = ,R ± 0 Offset
1	R	R	I	0	1	0	1	EA = ,R ± ACCB Offset
1	R	R	I	0	1	1	0	EA = ,R ± ACCA Offset
1	R	R	I	1	0	0	0	EA = ,R ± 7 Bit Offset
1	R	R	I	1	0	0	1	EA = ,R ± 15 Bit Offset
1	R	R	I	1	0	1	1	EA = ,R ± D Offset
1	X	X	I	1	1	0	0	EA = ,PC ± 7 Bit Offset
1	X	X	I	1	1	0	1	EA = ,PC ± 15 Bit Offset
1	R	R	1	1	1	1	1	EA = ,Address

Addressing Mode Field

Indirect Field
Sign bit when B7 = 0

Register Field
00:R = X
01:R = Y
10:R = U
11:R = S
X = Don't Care

Zero-Offset Indexed — In this mode, the selected pointer register contains the effective address of the data to be used by the instruction. This is the fastest indexing mode.

Examples are:

LDD 0,X
LDA 0,S

Constant Offset Indexed — In this mode a two's-complement offset and the contents of one of the pointer registers are added to form the effective address of the operand. The pointer register's initial content is unchanged by the addition.

Three sizes of offsets are available:

± 4-bit (-16 to +15)
± 7-bit (-128 to +127)
± 15-bit (-32768 to +32767)

The two's complement 5-bit offset is included in the postbyte and therefore is most efficient in use of bytes and cycles. The two's complement 8-bit offset is contained in a single byte following the postbyte. The two's complement 16-bit offset is in the two bytes following the postbyte. In most cases the programmer need not be concerned with the size of this offset since the assembler will select the optional size automatically.

Examples of constant-offset indexing are:

LDA 23,X
LDX -2,S
LDY 300,X
LDU CAT,Y

TABLE 2 — INDEXED ADDRESSING MODES

		Non Indirect				Indirect			
Type	Forms	Assembler Form	Postbyte OP Code	+~	+#	Assembler Form	Postbyte OP Code	+~	+#
Constant Offset From R (Signed Offsets)	No Offset	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5 Bit Offset	n, R	0RRnnnnn	1	0	defaults to 8-bit			
	8 Bit Offset	n, R	1RR01000	1	1	[n, R]	1RR11000	4	1
	16 Bit Offset	n, R	1RR01001	4	2	[n, R]	1RR11001	7	2
Accumulator Offset From R (Signed Offsets)	A — Register Offset	A, R	1RR00110	1	0	[A, R]	1RR10110	4	0
	B — Register Offset	B, R	1RR00101	1	0	[B, R]	1RR10101	4	0
	D — Register Offset	D, R	1RR01011	4	0	[D, R]	1RR11011	7	0
Auto Increment/Decrement R	Increment By 1	,R+	1RR00000	2	0	not allowed			
	Increment By 2	,R++	1RR00001	3	0	[,R++]	1RR10001	6	0
	Decrement By 1	,-R	1RR00010	2	0	not allowed			
	Decrement By 2	,--R	1RR00011	3	0	[,--R]	1RR10011	6	0
Constant Offset From PC	8 Bit Offset	n, PCR	1XX01100	1	1	[n, PCR]	1XX11100	4	1
	16 Bit Offset	n, PCR	1XX01101	5	2	[n, PCR]	1XX11101	8	2
Extended Indirect	16 Bit Address	—	—	—	—	[n]	10011111	5	2

R = X, Y, U or S   X = 00   Y = 01
X = Don't Care   U = 10   S = 11

+~ and +# Indicate the number of additional cycles and bytes for the particular variation.

Accumulator-Offset Indexed — This mode is similar to constant offset indexed except that the two's-complement value in one of the accumulators (A, B or D) and the content of one of the pointer registers are added to form the effective address of the operand. The contents of both the accumulator and the pointer register are unchanged by the addition. The postbyte specifies which accumulator to use as an offset and no additional bytes are required. The advantage of an accumulator offset is that the value of the offset can be calculated by a program at run-time.

Some examples are:

```
LDA B,Y
LDX D,Y
LEAX B,X
```

Auto Increment/Decrement Indexed — In the auto increment addressing mode, the pointer register contains the address of the operand. Then, after the pointer register is used it is incremented by one or two. This addressing mode is useful in stepping through tables, moving data, or for the creation of software stacks. In auto decrement, the pointer register is decremented prior to use as the address of the data. The use of auto decrement is similiar to that of auto increment but the tables, etc. are scanned from the high to low addresses. The size of the increment/decrement can be either one or two to allow for tables of either 8 or 16-bit data to be accessed and is selectable by the programmer. The pre-decrement, post-increment nature of these modes allow them to be used to create additional software stacks that behave identically to the U and S stacks.

Some examples of the auto increment/decrement addressing modes are:

```
LDA ,X+
STD ,Y++
LDB ,-Y
LDX ,--S
```

INDEXED INDIRECT

All of the indexing modes with the exception of auto increment/decrement by one, or a ± 4-bit offset may have an additional level of indirection specified. In Indirect addressing, the effective address is contained at the location specified by the content of the Index register plus any offset. In the example below, the A accumulator is loaded indirectly using an effective address calculated from the Index register and an offset.

```
                     Before Execution
                    A = XX (don't care)
                        X = $F000
$0100    LDA      [10, X]    EA is now $F010

$F010    $F1                 F150 is now the
$F011    $50                 new EA

$F150    $AA
                      After Execution
                 A = $AA Actual Data Loaded
```

All modes of indexed indirect are included except those which are meaningless (e.g. auto increment/decrement by 1 indirect). Some examples of indexed indirect are:

```
LDA   [,X]
LDD   [10,S]
LDA   [B,Y]
LDD   [,X++]
```

RELATIVE ADDRESSING

The byte(s) following the branch opcode is (are) treated as a signed offset which is added to the program counter. If the branch condition is true then the calculated address (PC + signed offset) is loaded into the program counter. Program execution continues at the new location as indicated by the PC; Short (1 byte offset) and long (2 bytes offset) relative addressing modes are available. All of memory can be reached in long relative addressing as an effective address is interpreted modulo 2^{16}. Some examples of relative addressing are:

```
          BEQ     CAT        (short)
          BGT     DOG        (short)
CAT       LBEQ    RAT        (long)
DOG       LBGT    RABBIT     (long)
           •
           •
           •
RAT       NOP
RABBIT    NOP
```

PROGRAM COUNTER RELATIVE

The PC can be used as the pointer register with 8 or 16-bit signed offsets. As in relative addressing the offset is added to the current PC to create the effective address. The effective address is then used as the address of the operand or data. Program Counter Relative Addressing is used for writing position independent programs. Tables related to a particular routine will maintain the same relationship after the routine is moved, if referenced relative to the Program Counter. Examples are:

```
LDA    CAT, PCR
LEAX   TABLE, PCR
```

Since program counter relative is a type of indexing, an additional level of indirection is available.

```
LDA   [CAT, PCR]
LDU   [DOG, PCR]
```

MC6809 INSTRUCTION SET

The instruction set of the MC6809 is similar to that of the MC6800 and is upward compatible at the source code level. The number of opcodes has been reduced from 72 to 59, but because of the expanded architecture and additional addressing modes, the number of available opcodes (with different addressing modes) has risen from 197 to 1464.

Some of the new instructions and addressing modes are described in detail below:

PSHU/PSHS

The push instructions have the capability of pushing onto either the hardware stack (S) or user stack (U) any or all of the MPU registers with a single instruction.

PULU/PULS

The pull instructions have the same capability of the push instruction, in reverse order. The byte immediate following the push or pull opcode determines which register or registers are to be pushed or pulled. The actual PUSH/PULL sequence is fixed; each bit defines a unique register to push or pull as shown in Figure 16.

TFR/EXG

Within the MC6809, any register may be transferred to or exchanged with another of like-size, i.e. 8-bit to 8-bit or 16-bit to 16-bit. Bits 4-7 of postbyte define the source register, while bits 0-3 represent the destination register. These are denoted as follows:

0000	— D	0101	— PC
0001	— X	1000	— A
0010	— Y	1001	— B
0011	— U	1010	— CC
0100	— S	1011	— DP

Note: All other combinations are undefined and INVALID.

Load Effective Address

The LEA works by calculating the effective address used in an indexed instruction and stores that address value, rather than the data at that address, in a pointer register. This makes all the features of the internal addressing hardware available to the programmer. Some of the implications of this instruction are illustrated in the following table of examples:

The LEA instruction also allows the user to access data in a position independent manner. For example:

```
        LEAX    MSG1, PCR
        LBSR    PDATA (Print message routine)


MSG1 FCC        'MESSAGE'
```

FIGURE 16 — PUSH/PULL POSTBYTE

```
           ← Pull Order   Push Order →
PC    U    Y    X    DP    B    A    CC      PSHS/PULS
    FFFF ...... ← increasing memory address → ...... 0000
PC    S    Y    X    DP    B    A    CC      PSHU/PULU
```

TABLE 3 — LEA EXAMPLES

Instruction		Operation		Comment
LEAX	10, X	X + 10	→ X	Adds 5-bit constant 10 to X
LEAX	500, X	X + 500	→ X	Adds 16-bit constant 500 to X
LEAY	A, Y	Y + A	→ Y	Adds 8-bit accumulator to Y
LEAY	D, Y	Y + D	→ Y	Adds 16-bit D accumulator to Y
LEAU	-10, U	U - 10	→ U	Subtracts 10 from U
LEAS	-10, S	S - 10	→ S	Used to reserve area on stack
LEAS	10, S	S + 10	→ S	Used to 'clean up' stack
LEAX	5, S	S + 5	→ X	Transfers as well as adds

This sample program prints "message". By writing MSG1,PCR, the assembler computes the distance between the present address and MSG1. This result is placed as a constant into the LEAX instruction which will be indexed from the PC value at the time of execution. No matter where the code is located, when it is executed, the computed offset from the PC will put the absolute address of MSG1 into the X pointer register. This code is totally position independent.

MUL

Multiplies the unsigned binary numbers in the A and B accumulator and places the unsigned result into the 16-bit D accumulator. This unsigned multiply also allows multiple-precision multiplications.

Long And Short Relative Branches

The MC6809 has the capability of program counter relative branching throughout the entire memory map. In this mode, if the branch is to be taken, the 8 or 16-bit signed offset is added to the value of the program counter to be used as the effective address. This allows the program to branch anywhere in the 64K memory map. Position independent code can be easily generated through the use of relative branching. Both short (8-bit) and long (16-bit) branches are available.

SYNC

After encountering a Sync instruction, the MPU enters a Sync state, stops processing instructions and waits for an interrupt. If the pending interrupt is non-maskable (NMI) or maskable (FIRQ, IRQ) with its mask bit (F or I) clear, the processor will clear the Sync state and perform the normal interrupt stacking and service routine. Since FIRQ and IRQ are not edge-triggered, a low level with a minimum duration of three cycles is required to assure that the interrupt will be taken. If the pending interrupt is maskable (FIRQ, IRQ) with its mask bit (F or I) set, the processor will clear the Sync state and continue processing in sequence. Figure 18 depicts Sync timing.

Software Interrupts

A Software Interrupt is an instruction which will cause an interrupt, and its associated vector fetch. These Software Interrupts are useful in operating system calls, software debugging, trace operations, memory mapping, and software development systems. Three levels of SWI are available on this MC6809, and are prioritized in the following order: SWI, SWI2, SWI3.

16-Bit Operations

The MC6809 has the capability of processing 16-bit data. These instructions include loads, stores, compares, adds, subtracts, transfers, exchanges, pushes and pulls.

CYCLE-BY-CYCLE OPERATION

The address bus cycle-by-cycle performance chart illustrates the memory-access sequence corresponding to each possible instruction and addressing mode in the MC6809. Each instruction begins with an opcode fetch. While that opcode is being internally decoded, the next program byte is always fetched. (Most instructions will use the next byte, so this technique cnsiderably speeds throughput). Next, the operation of each opcode will follow the flow chart. $\overline{\text{VMA}}$ is an indication of FFFF_{16} on the address bus, $R/\overline{W} = 1$ and $BS = 0$. The following examples illustrate the use of the chart; see Figure 19.

LBSR (Branch taken)

Cycle #	
1	opcode Fetch
2	opcode +
3	opcode +
4	$\overline{\text{VMA}}$
5	VMA
6	ADDR
7	$\overline{\text{VMA}}$
8	STACK (write)
9	STACK (write)

DEC (Extended)

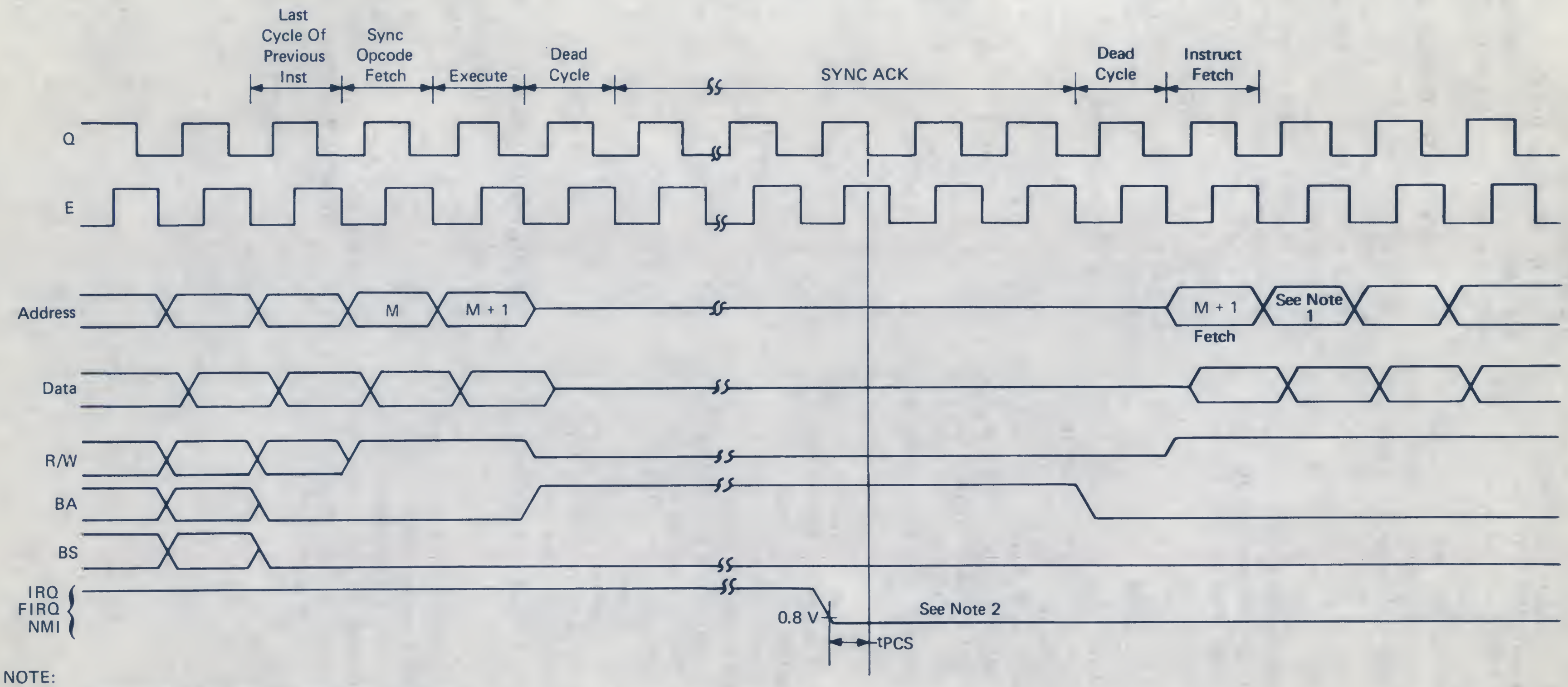
1	opcode Fetch
2	opcode +
3	opcode +
4	$\overline{\text{VMA}}$
5	ADDR (read)
6	$\overline{\text{VMA}}$
7	ADDR (write)

MC6809 INSTRUCTION SET TABLES

The instructions of the MC6809 have been broken down into five different categories. They are as follows:

8-Bit operation (Table 4)
16-Bit operation (Table 5)
Index register/stack pointer instructions (Table 6)
Relative branches (long and short) (Table 7)
Miscellaneous instructions (Table 8)
Hexadecimal Value instructions (Table 9)

FIGURE 18 — SYNC TIMING



NOTE:
1. If the mask bit is set when the interrupt is requested processing will continue with instruction execution fetched from previous step. However, if an NMI or an unmasked FIRQ or IRQ caused interrupt, the address placed on bus from previous cycle (M + 1) remains on bus and processing continues with this cycle as (m + 1) or (n + 1) of interrupt timing.
2. If mask bits are clear IRQ & FIRQ must be held low for three cycles to guarantee interrupt to be taken, although only one cycle is necessary to bring the processor out of SYNC.

FIGURE 19 — ADDRESS BUS CYCLE-BY-CYCLE PERFORMANCE

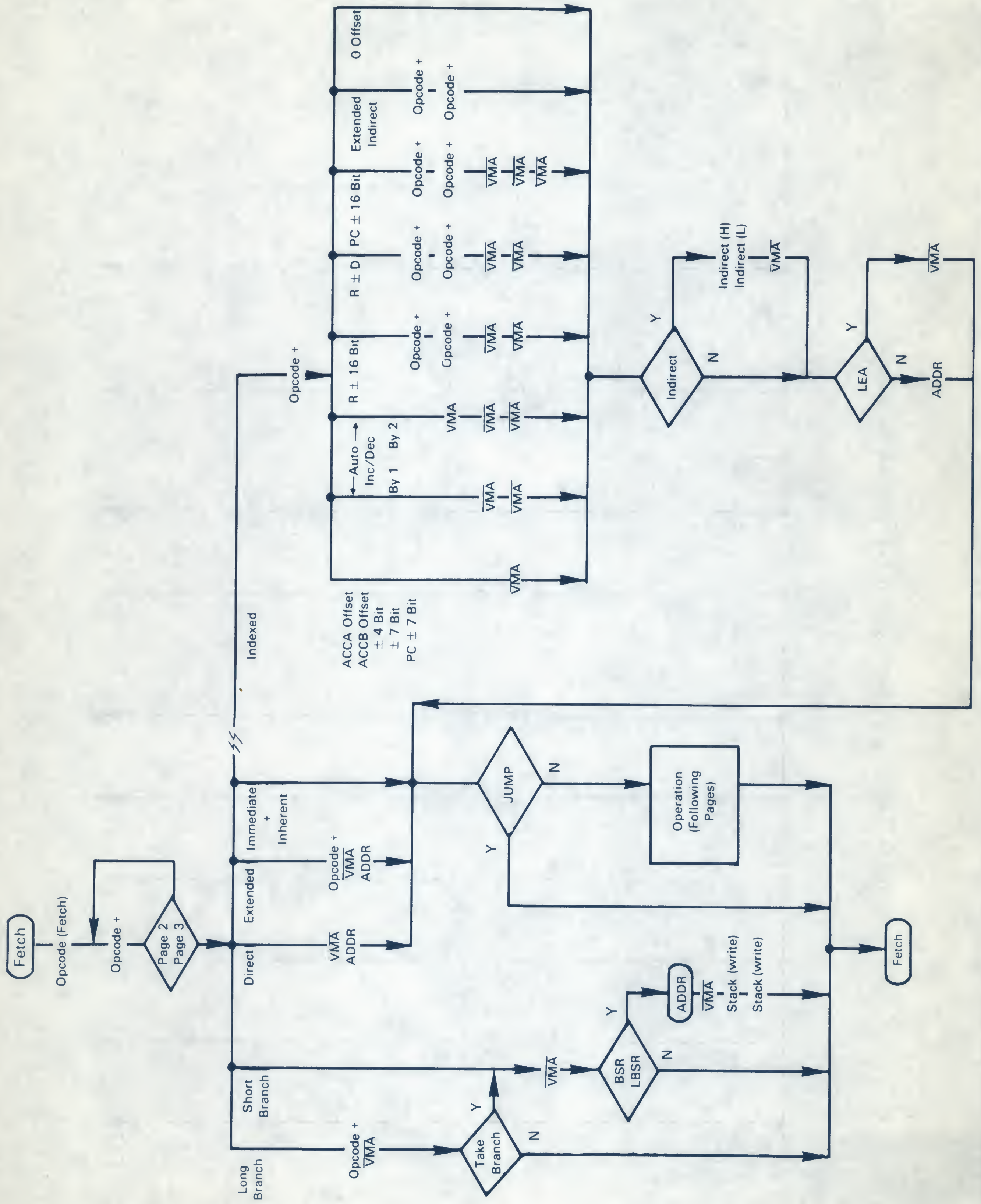
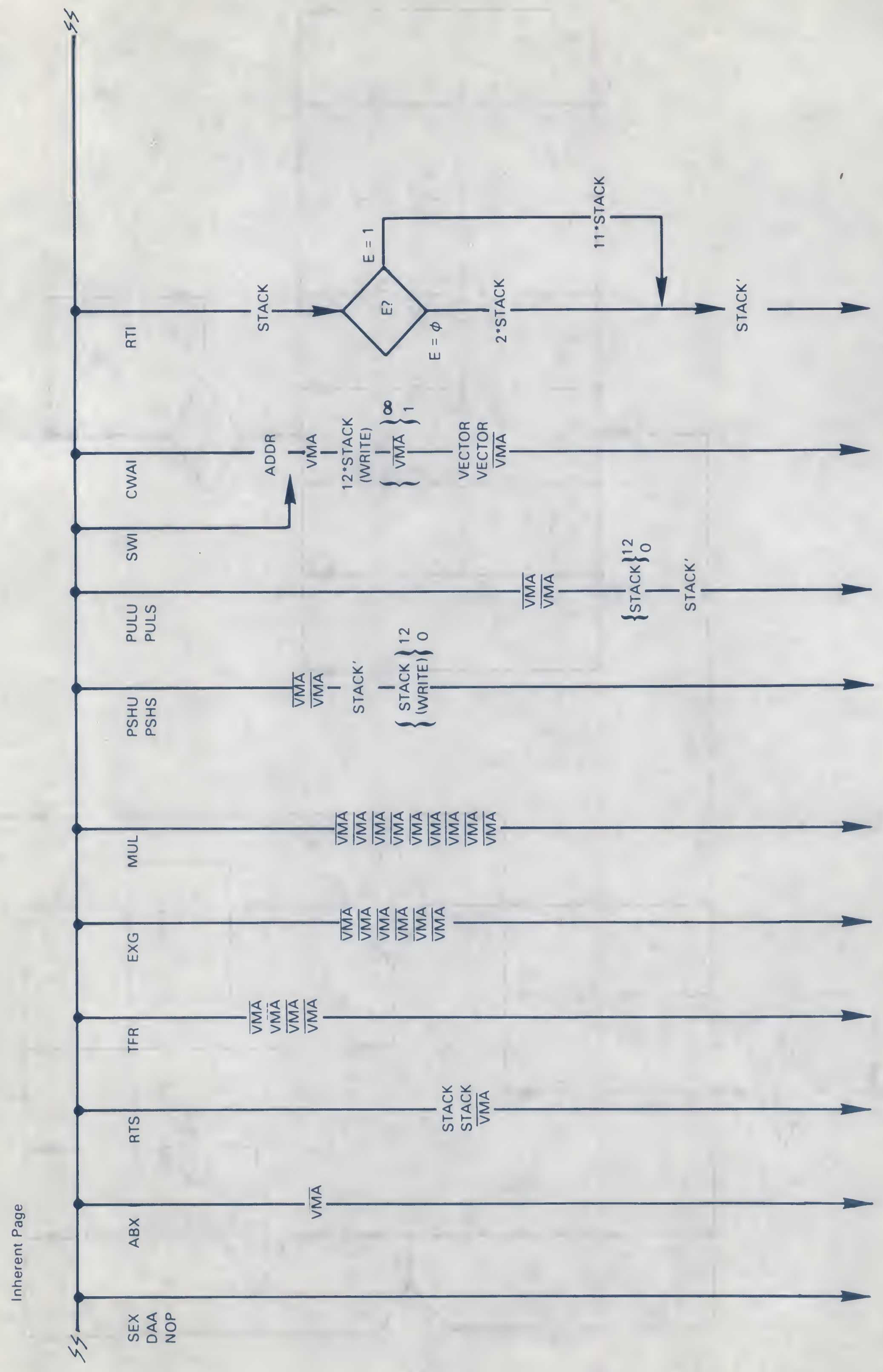


FIGURE 19 — ADDRESS BUS CYCLE-BY-CYCLE PERFORMANCE (CONT.)



MOTOROLA Semiconductor Products Inc.

FIGURE 19 — ADDRESS BUS CYCLE-BY-CYCLE PERFORMANCE (CONT.)

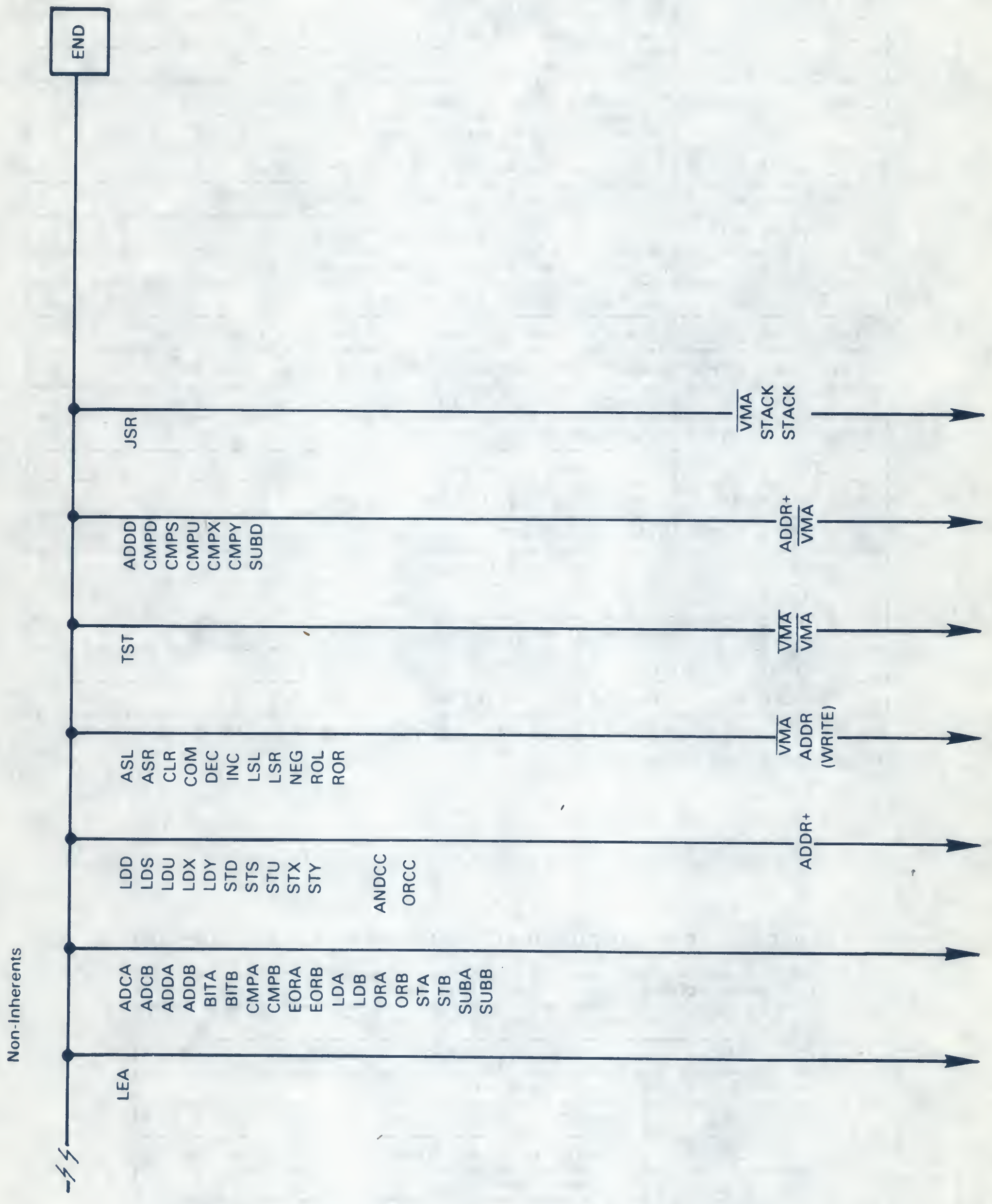

Non-Inherents
LEA
ADCA ADCB ADDA ADDB BITA BITB CMPA CMPB EORA EORB LDA LDB ORA ORB STA STB SUBA SUBB
LDD LDS LDU LDX LDY STD STS STU STX STY
ANDCC ORCC
ADDR+
ASL ASR CLR COM DEC INC LSL LSR NEG ROL ROR
$\overline{VMA}$ ADDR (WRITE)
TST
$\overline{VMA}$ $\overline{VMA}$
ADDD CMPD CMPS CMPU CMPX CMPY SUBD
ADDR+ $\overline{VMA}$
JSR
$\overline{VMA}$ STACK STACK
END

TABLE 4 — 8-BIT ACCUMULATOR AND MEMORY INSTRUCTIONS

Mnemonic(s)	Operation
ADCA, ADCB	Add memory to accumulator with carry
ADDA, ADDB	Add memory to accumulator
ANDA, ANDB	And memory with accumulator
ASL, ASLA, ASLB	Arithmetic shift of accumulator or memory left
ASR, ASRA, ASRB	Arithmetic shift of accumulator or memory right
BITA, BITB	Bit test memory with accumulator
CLR, CLRA, CLRB	Clear accumulator or memory location
CMPA, CMPB	Compare memory from accumulator
COM, COMA, COMB	Complement accumulator or memory location
DAA	Decimal adjust A-accumulator
DEC, DECA, DECB	Decrement accumulator or memory location
EORA, EORB	Exclusive or memory with accumulator
EXG R1, R2	Exchange R1 with R2 (R1, R2 = A, B, CC, DP)
INC, INCA, INCB	Increment accumulator or memory location
LDA, LDB	Load accumulator from memory
LSL, LSLA, LSLB	Logical shift left accumulator or memory location
LSR, LSRA, LSRB	Logical shift right accumulator or memory location
MUL	Unsigned multiply (A x B → D)
NEG, NEGA, NEGB	Negate accumulator or memory
ORA, ORB	Or memory with accumulator
ROL, ROLA, ROLB	Rotate accumulator or memory left
ROR, RORA, RORB	Rotate accumulator or memory right
SBCA, SBCB	Subtract memory from accumulator with borrow
STA, STB	Store accumulator to memory
SUBA, SUBB	Subtract memory from accumulator
TST, TSTA, TSTB	Test accumulator or memory location
TFR, R1, R2	Transfer R1 to R2 (R1, R2 = A, B, CC, DP)

NOTE: A, B, CC, or DP may be pushed to (pulled from) either stack with PSHS, PSHU, (PULS, PULU) instructions.

TABLE 5 — 16-BIT ACCUMULATOR AND MEMORY INSTRUCTIONS

Mnemonic(s)	Operation
ADDD	Add memory to D accumulator
CMPD	Compare memory from D accumulator
EXG D, R	Exchange D with X, Y, S, U or PC
LDD	Load D accumulator from memory
SEX	Sign Extend B accumulator into A accumulator
STD	Store D accumulator to memory
SUBD	Subtract memory from D accumulator
TFR D, R	Transfer D to X, Y, S, U or PC
TFR R, D	Transfer X, Y, S, U or PC to D

TABLE 6 — INDEX REGISTER/STACK POINTER INSTRUCTIONS

Mnemonic(s)	Operation
CMPS, CMPU	Compare memory from stack pointer
CMPX, CMPY	Compare memory from index register
EXG R1, R2	Exchange D, X, Y, S, U, or PC with D, X, Y, S, U or PC
LEAS, LEAU	Load effective address into stack pointer
LEAX, LEAY	Load effective address into index register
LDS, LDU	Load stack pointer from memory
LDX, LDY	Load index register from memory
PSHS	Push any register(s) onto hardware stack (except S)
PSHU	Push any register(s) onto user stack (except U)
PULS	Pull any register(s) from hardware stack (except S)
PULU	Pull any register(s) from hardware stack (except U)
STS, STU	Store stack pointer to memory
STX, STY	Store index register to memory
TFR R1, R2	Transfer D, X, Y, S, U or PC to D, X, Y, S, U or PC
ABX	Add B accumulator to X (unsigned)

TABLE 7 — BRANCH INSTRUCTIONS

Mnemonic(s)	Operation
BCC, LBCC	Branch if carry clear
BCS, LBCS	Branch if carry set
BEQ, LBEQ	Branch if equal
BGE, LBGE	Branch if greater than or equal (signed)
BGT, LBGT	Branch if greater (signed)
BHI, LBHI	Branch if higher (unsigned)
BHS, LBHS	Branch if higher or same (unsigned)
BLE, LBLE	Branch if less than or equal (signed)
BLO, LBLO	Branch if lower (unsigned)
BLS, LBLS	Branch if lower or same (unsigned)
BLT, LBLT	Branch if less than (signed)
BMI, LBMI	Branch if minus
BNE, LBNE	Branch if not equal
BPL, LBPL	Branch if plus
BRA, LBRA	Branch always
BRN, LBRN	Branch never
BSR, LBSR	Branch to subroutine
BVC, LBVC	Branch if overflow clear
BVS, LBVS	Branch if overflow set

TABLE 8 — MISCELLANEOUS INSTRUCTIONS

Mnemonic(s)	Operation
ANDCC	AND condition code register
CWAI	AND condition code register, then wait for interrupt
NOP	No operation
ORCC	OR condition code register
JMP	Jump
JSR	Jump to subroutine
RTI	Return from interrupt
RTS	Return from subroutine
SWI, SWI2, SWI3	Software interrupt (absolute indirect)
SYNC	Synchronize with interrupt line

TABLE 9 — HEXADECIMAL VALUES OF MACHINE CODES

OP Mnem	Mode	~	#
00 NEG	Direct	6	2
01 *			
02 *			
03 COM		6	2
04 LSR		6	2
05 *			
06 ROR		6	2
07 ASR		6	2
08 ASL/LSL		6	2
09 ROL		6	2
0A DEC		6	2
0B *			
0C INC		6	2
0D TST		6	2
0E JMP		3	2
0F CLR	Direct	6	2
10 Page 2	—	—	—
11 Page 3	—	—	—
12 NOP	Inherent	2	1
13 SYNC	Inherent	2	1
14 *			
15 *			
16 LBRA	Relative	5	3
17 LBSR	Relative	9	3
18 *			
19 DAA	Inherent	2	1
1A ORCC	Immed	3	2
1B *			
1C ANDCC	Immed	3	2
1D SEX	Inherent	2	1
1E EXG		8	2
1F TFR	Inherent	6	2
20 BRA	Relative	3	2
21 BRN		3	2
22 BHI		3	2
23 BLS		3	2
24 BHS/BCC		3	2
25 BLO/BCS		3	2
26 BNE		3	2
27 BEQ		3	2
28 BVC		3	2
29 BVS		3	2
2A BPL		3	2
2B BMI		3	2
2C BGE		3	2
2D BLT		3	2
2E BGT		3	2
2F BLE	Relative	3	2

OP Mnem	Mode	~	#
30 LEAX	Indexed	4+	2+
31 LEAY		4+	2+
32 LEAS		4+	2+
33 LEAU	Indexed	4+	2+
34 PSHS	Inherent	5+	2
35 PULS		5+	2
36 PSHU		5+	2
37 PULU		5+	2
38 *			
39 RTS		5	1
3A ABX		3	1
3B RTI		6/15	1
3C CWAI		20	2
3D MUL		11	1
3E *			
3F SWI	Inherent	19	1
40 NEGA	Inherent	2	1
41 *			
42 *			
43 COMA		2	1
44 LSRA		2	1
45 *			
46 RORA		2	1
47 ASRA		2	1
48 ASLA/LSLA		2	1
49 ROLA		2	1
4A DECA		2	1
4B *			
4C INCA		2	1
4D TSTA		2	1
4E *			
4F CLRA	Inherent	2	1
50 NEGB	Inherent	2	1
51 *			
52 *			
53 COMB		2	1
54 LSRB		2	1
55 *			
56 *			
56 RORB		2	1
57 ASRA		2	1
58 ASLB/LSLB		2	1
59 ROLB		2	1
5A DECB		2	1
5B *			
5C INCB		2	1
5D TSTB		2	1
5E *			
5F CLRB	Inherent	2	1

OP Mnem	Mode	~	#
60 NEG	Indexed	6+	2+
61 *			
62 *			
63 COM		6+	2+
64 LSR		6+	2+
65 *			
66 ROR		6+	2+
67 ASR		6+	2+
68 ASL/LSL		6+	2+
69 ROL		6+	2+
6A DEC		6+	2+
6B *			
6C INC		6+	2+
6D TST		6+	2+
6E JMP		3+	2+
6F CLR	Indexed	6+	2+
70 NEG	Extended	7	3
71 *			
72 *			
73 COM		7	3
74 LSR		7	3
75 *			
76 ROR		7	3
77 ASR		7	3
78 ASL/LSL		7	3
79 ROL		7	3
7A DEC		7	3
7B *			
7C INC		7	3
7D TST		7	3
7E JMP		4	3
7F CLR	Extended	7	3
80 SUBA	Immed	2	2
81 CMPA		2	2
82 SBCA		2	2
83 SUBD		4	3
84 ANDA		2	2
85 BITA		2	2
86 LDA		2	2
87 *			
88 EORA		2	2
89 ADCA		2	2
8A ORA		2	2
8B ADDA		2	2
8C CMPX	Immed	4	3
8D BSR	Relative	7	2
8E LDX	Immed	3	3
8F *			

Legend:

~ Number of MPU cycles (less possible push/pull or indexed-mode cycles)

# Number of program bytes

* Denotes unused opcode

TABLE 9 – HEXADECIMAL VALUES OF MECHANICAL CODES (CONTINUED)

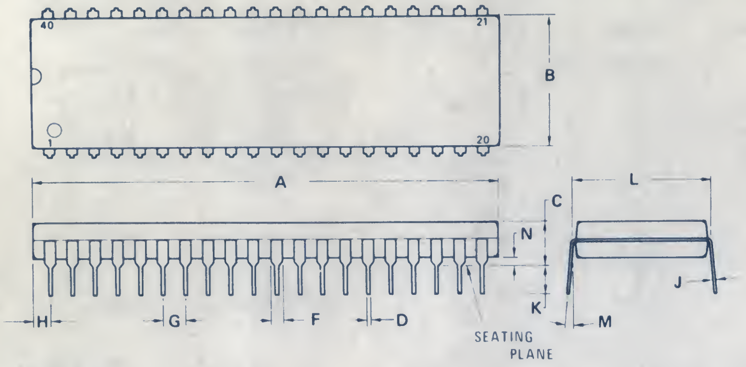
OP Mnem	Mode	~	#
90 SUBA	Direct	4	2
91 CMPA		4	2
92 SBCA		4	2
93 SUBD		6	2
94 ANDA		4	2
95 BITA		4	2
96 LDA		4	2
97 STA		4	2
98 EORA		4	2
99 ADCA		4	2
9A ORA		4	2
9B ADDA		4	2
9C CMPX		6	2
9D JSR		7	2
9E LDX		5	2
9F STX	Direct	5	2
A0 SUBA	Indexed	4+	2+
A1 CMPA		4+	2+
A2 SBCA		4+	2+
A3 SUBD		6+	2+
A4 ANDA		4+	2+
A5 BITA		4+	2+
A6 LDA		4+	2+
A7 STA		4+	2+
A8 EORA		4+	2+
A9 ADCA		4+	2+
AA ORA		4+	2+
AB ADDA		4+	2+
AC CMPX		6+	2+
AD JSR		7+	2+
AE LDX		5+	2+
AF STX	Indexed	5+	2+
B0 SUBA	Extended	5	3
B1 CMPA		5	3
B2 SBCA		5	3
B3 SUBD		7	3
B4 ANDA		5	3
B5 BITA		5	3
B6 LDA		5	3
B7 STA		5	3
B8 EORA		5	3
B9 ADCA		5	3
BA ORA		5	3
BB ADDA		5	3
BC CMPX		7	3
BD JSR		8	3
BE LDX		6	3
BF STX	Extended	6	3
C0 SUBB	Immed	2	2
C1 CMPB		2	2
C2 SBCB		2	2
C3 ADDD		4	3
C4 ANDB		2	2
C5 BITB	Immed	2	2

OP Mnem	Mode	~	#
C6 LDB	Immed	2	2
C7 *			
C8 EORB		2	2
C9 ADCB		2	2
CA ORB		2	2
CB ADDB		2	2
CC LDD		3	3
CD *			
CE LDU	Immed	3	3
CF *			
D0 SUBB	Direct	4	2
D1 CMPB		4	2
D2 SBCB		4	2
D3 ADDD		6	2
D4 ANDB		4	2
D5 BITB		4	2
D6 LDB		4	2
D7 STB		4	2
D8 EORB		4	2
D9 ADCB		4	2
DA ORB		4	2
DB ADDB		4	2
DC LDD		5	2
DD STD		5	2
DE LDU		5	2
DF STU	Direct	5	2
E0 SUBB	Indexed	4+	2+
E1 CMPB		4+	2+
E2 SBCB		4+	2+
E3 ADDD		6+	2+
E4 ANDB		4+	2+
E5 BITB		4+	2+
E6 LDB		4+	2+
E7 STB		4+	2+
E8 EORB		4+	2+
E9 ADCB		4+	2+
EA ORB		4+	2+
EB ADDB		4+	2+
EC LDD		5+	2+
ED STD		5+	2+
EE LDU		5+	2+
EF STU	Indexed	5+	2+
F0 SUBB	Extended	5	3
F1 CMPB		5	3
F2 SBCB		5	3
F3 ADDD		7	3
F4 ANDB		5	3
F5 BITB		5	3
F6 LDB		5	3
F7 STB		5	3
F8 EORB		5	3
F9 ADCB		5	3
FA ORB		5	3
FB ADDB	Extended	5	3

OP Mnem	Mode	~	#
FC LDD	Extended	6	3
FD STD		6	3
FE LDU		6	3
FF STU	Extended	6	3

OP	Mnem	Mode	~	#
1021	LBRN	Relative	5	4
1022	LBHI		5(6)	4
1023	LBLS		5(6)	4
1024	LBHS/LBCC		5(6)	4
1025	LBCS/LBLO		5(6)	4
1026	LBNE		5(6)	4
1027	LBEQ		5(6)	4
1028	LBVC		5(6)	4
1029	LBVS		5(6)	4
102A	LBPL		5(6)	4
102B	LBMI		5(6)	4
102C	LBGE		5(6)	4
102D	LBLT	Relative	5(6)	4
102E	LBGT	Relative	5(6)	4
102F	LBLE	Relative	5(6)	4
103F	SWI/2	Inherent	20	2
1083	CMPD	Immed	5	4
108C	CMPY		5	4
108E	LDY	Immed	4	4
1093	CMPD	Direct	7	3
109C	CMPY		7	3
109E	LDY		6	3
109F	STY	Direct	6	3
10A3	CMPD	Indexed	7+	3+
10AC	CMPY		7+	3+
10AE	LDY		6+	3+
10AF	STY	Indexed	6+	3+
10B3	CMPD	Extended	8	4
10BC	CMPY		8	4
10BE	LDY		7	4
10BF	STY	Extended	7	4
10CE	LDS	Immed	4	4
10DE	LDS	Direct	6	3
10DF	STS	Direct	6	3
10EE	LDS	Indexed	6+	3+
10EF	STS	Indexed	6+	3+
10FE	LDS	Extended	7	4
10FF	STS	Extended	7	4
113F	SWI/3	Inherent	20	2
1183	CMPU	Immed	5	4
118C	CMPS	Immed	5	4
1193	CMPU	Direct	7	3
119C	CMPS	Direct	7	3
11A3	CMPU	Indexed	7+	3+
11AC	CMPS	Indexed	7+	3+
11B3	CMPU	Extended	8	4
11BC	CMPS	Extended	8	4

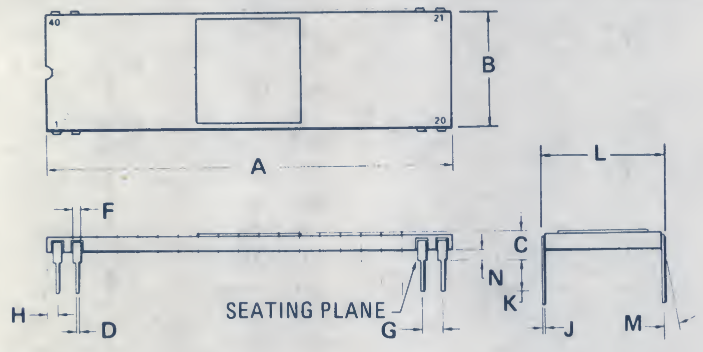
NOTE: All unused opcodes are both undefined and illegal.



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	51.69	52.45	2.035	2.065
B	13.72	14.22	0.540	0.560
C	3.94	5.08	0.155	0.200
D	0.36	0.56	0.014	0.022
F	1.02	1.52	0.040	0.060
G	2.54 BSC		0.100 BSC	
H	1.65	2.16	0.065	0.085
J	0.20	0.38	0.008	0.015
K	2.92	3.43	0.115	0.135
L	15.24 BSC		0.600 BSC	
M	0°	15°	0°	15°
N	0.51	1.02	0.020	0.040

NOTES:
1. POSITIONAL TOLERANCE OF LEADS (D), SHALL BE WITHIN 0.25 mm (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION B DOES NOT INCLUDE MOLD FLASH.

P SUFFIX
PLASTIC PACKAGE
CASE 711-03



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	50.29	51.31	1.980	2.020
B	14.94	15.34	0.588	0.604
C	3.05	4.06	0.120	0.160
D	0.38	0.53	0.015	0.021
F	0.76	1.40	0.030	0.055
G	2.54 BSC		0.100 BSC	
H	0.76	1.78	0.030	0.070
J	0.20	0.33	0.008	0.013
K	2.54	4.19	0.100	0.165
L	14.99	15.49	0.590	0.610
M	–	10°	–	10°
N	1.02	1.52	0.040	0.060

NOTES:
1. LEADS, TRUE POSITIONED WITHIN 0.25 mm (0.010) DIA (AT SEATING PLANE), AT MAX MAT'L CONDITION.
2. DIMENSION "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.

L SUFFIX
CERAMIC PACKAGE
CASE 715-03

MOTOROLA *Semiconductor Products Inc.*